What is being transferred in transfer learning?

Behnam Neyshabur[*,*]
Google
neyshabur@google.com

Hanie Sedghi[*]
Google Brain
hsedghi@google.com

Chiyuan Zhang[*]
Google Brain
chiyuan@google.com

August 27, 2020

Abstract

One desired capability for machines is the ability to transfer their knowledge of one domain to another where data is (usually) scarce. Despite ample adaptation of transfer learning in various deep learning applications, we yet do not understand what enables a successful transfer and which part of the network is responsible for that. In this paper, we provide new tools and analyses to address these fundamental questions. Through a series of analyses on transferring to block-shuffled images, we separate the effect of feature reuse from learning low-level statistics of data and show that some benefit of transfer learning comes from the latter. We present that when training from pre-trained weights, the model stays in the same basin in the loss landscape and different instances of such model are similar in feature space and close in parameter space.

1 Introduction

One desired capability of machines is to transfer their knowledge of a domain it is trained on (the source domain) to another domain (the target domain) where data is (usually) scarce or a fast training is needed. There has been a plethora of works on using this framework in different applications such as object detection [13, 38], image classification [40, 29, 24] and segmentation [5, 17], and various medical imaging tasks [42, 37, 6], to name a few. In some cases there is nontrivial difference in visual forms between the source and the target domain [42, 37]. However, we yet do not understand what enables a successful transfer and which parts of the network are responsible for that. In this paper we address these fundamental questions.

To ensure that we are capturing a general phenomena, we look into target domains that are intrinsically different and diverse. We use CheXpert [21] which is a medical imaging dataset of chest x-rays considering different diseases. We also consider DomainNet [33] datasets that are specifically designed to probe transfer learning in diverse domains. The domains range from real images to sketches, clipart and painting samples. See Figure 1 for sample images of the transfer learning tasks studied in this paper.

We use a series of analysis to answer the question of what is being transferred. First, we investigate feature-reuse by shuffling the data. In particular, we partition the image of the downstream tasks into equal sized blocks and shuffle the blocks randomly. The shuffling of blocks disrupts visual features in the images, especially with small block sizes (see Figure 1 for examples of shuffled images). We note the importance of feature re-use and we

[*]Equal contribution. Authors ordered randomly.

Before we move on to discuss the consequences of shuffling the image using blocks of different sizes, there are two very important questions that you should ask yourself to check if you have a good intuition about how CNNs work:

- Conv blocks make features translation invariant. Because we are just changing the position of the blocks in an image and not the features, will it really affect the final output?
- As we go deep down in the network, we rely more and more on high-level features. Changing blocks changes the spatial structure but does that change the high-level features? Will it affect the final results in any sense? Why/why not?

Same image. We divide the image into blocks of size 8x8 and shuffle them on the canvas

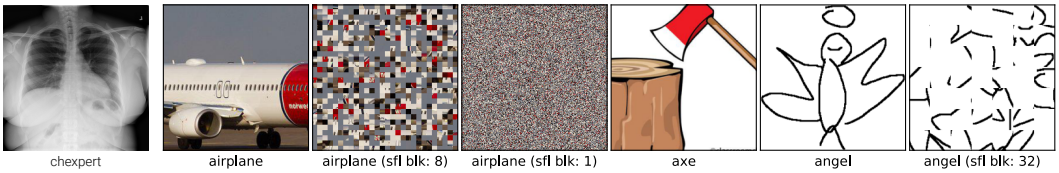


Figure 1: Sample images of dataset used for transfer learning downstream tasks. Left most: an example from CheXpert. The next three: an example from the DomainNet `real` dataset, the same image with random shuffling of 8×8 blocks and 1×1 blocks, respectively. The last three: examples from DomainNet `clipart` and `quickdraw`, and a 32×32 block-shuffled version of the `quickdraw` example.

also see that it is not the only role player in successful transfer. This experiment shows that low-level statistics of the data that is not disturbed by shuffling the pixels also play a role in successful transfer (Section 3.1). (1)

Next, we compare the detailed behaviors of trained models. In particular, we investigate the agreements/disagreements between models that are trained from pre-training versus scratch. We note that two instances of models that are trained from pre-trained weights make similar mistakes. However, if we compare these two classes of models, they have fewer common mistakes. This suggests that two instances of models trained from pre-trained weights are more similar in feature space compared to ones trained from random initialization. To investigate this further, we look into feature similarity, measured by *centered kernel alignment* (CKA) [26], at different modules[1] of the two model instances and observe that this is in fact the case. we also look into the ℓ_2 distance between parameters of the models and note that two instances of models that are trained from pre-trained weights are much closer in ℓ_2 distance compared to the ones trained from random initialization (Section 3.2). (2) (3) (4)

We then investigate the loss landscape of models trained from pre-training and random initialization weights and observe that there is no performance barrier between the two instances of models trained from pre-trained weights, which suggests that the pre-trained weights guide the optimization to a flat basin of the loss landscape. On the other hand, barriers are clearly observed between the solutions from two instances trained from randomly initialized weights, even when *the same* random weights are used for initialization (Section 3.3). (5) (6)

The next question is, can we pinpoint where feature reuse is happening? As shown in [45] different modules of the network have different robustness to parameter perturbation. Chatterji et al. [4] analyzed this phenomena and captured it under a module criticality measure. We extend this investigation to transfer learning with an improved definition of module criticality. Using this extended notion, we analyze criticality of different modules and observe that higher layers in the network have tighter valleys which confirms previous observations [44, 36] on features becoming more specialized as we go through the network and feature-reuse is happening in layers that are closer to the input. In addition, we observe that models that are trained from random initialization have a transition point in their valleys, which may be due to changing basins through training. (7)

Finally, inspired by our findings about the basin of loss landscape, we look into different checkpoint in the training of the pre-trained model and show that one can start fine-tuning from the earlier checkpoints without losing accuracy in the target domain (see Section 3.5). (8)

Our main contributions and takeaways are summarized below:

1. For a successful transfer both feature-reuse and low-level statistics of the data are important.

[1] A module is a node in the computation graph that has incoming edges from other modules and outgoing edges to other nodes and performs a linear transformation on its inputs. For layered architectures such as VGG, each layer is a module. For ResNets, a module can be a *residual block*, containing multiple layers and a skip connection.

- According to the authors, two models with pretrained weights make similar mistakes. Does that mean Mobilenet and ResNets make similar mistakes in transfer learning? Or is it something else? 🧐🧐🧐
- Two fine-tuned models have parameters much closer in the Euclidean space as compared to two models trained from random initialization. This makes sense
- Feature reuse happens in the layers that are closest to the input. This is intuitive because as we go deep, the network starts to look at more and more fine-grained high-level features.

- The authors say that we can start from an early checkpoint for fine-tuning and still achieve the same accuracy. Though I believe this is true as you can always get to the optimal point, but an early checkpoint would take more steps to get to the point as compared to starting with the best checkpoint. Though some of the latest checkpoints would have similar performance and in that case, it shouldn't matter much

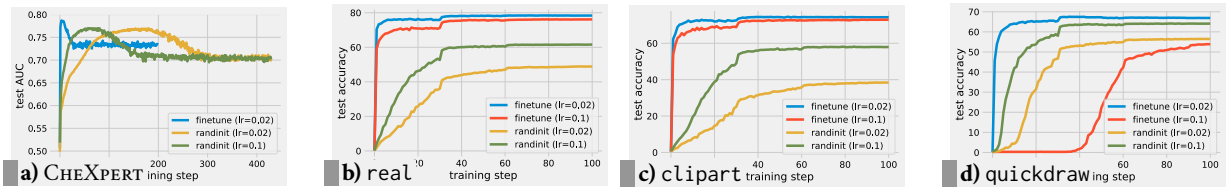


Figure 2: Learning curves comparing random initialization (RI-T) and finetuning from ImageNet pre-trained weights(P-T). For CheXpert, finetune with base learning rate 0.1 is not shown as it failed to converge.

2. Models trained from pre-trained weights make similar mistakes on target domain, have similar features and are surprisingly close in ℓ_2 distance in the parameter space. They are in the same basins of the loss landscape; while models trained from random initialization do not live in the same basin, make different mistakes, have different features and are farther away in ℓ_2 distance in the parameter space.

3. Modules in the lower layers are in charge of general features and modules in higher layers are more sensitive to perturbation of their parameters.

4. One can start from earlier checkpoints of pre-trained model without losing accuracy of the fine-tuned model. The starting point of such phenomena depends on when the pre-train model enters its final basin.

2 Problem Formulation and Setup

In transfer learning, we train a model on the source domain and then try to modify it to give us good predictions on the target domain. In order to investigate transfer learning, we analyze networks in four different cases: the pre-trained network, the network at random initialization, the network that is fine-tuned on target domain after pre-training on source domain and the model that is trained on target domain from random initialization. Since we will be referring to these four cases frequently, we use the following notations. T: trained, P: Pre-trained, RI: random initialization. Therefore we use the following abbreviations for the four models throughout the paper: RI (random initialization), P (pre-trained model), RI-T (model trained on target domain from random initialization), P-T (model trained/fine-tuned on target domain starting from pre-trained weights). We use ImageNet [7] pre-training for its prevalence in the community and consider CheXpert [21] and three sets from DomainNet [33] as downstream transfer learning tasks. See Appendix A for details about the experiment setup.

3 What is being transferred?

3.1 Role of feature reuse

Human visual system is compositional and hierarchical: neurons in the primary visual cortex (V1) respond to low level features like edges, while upper level neurons (e.g. the grandmother cell [15]) respond to complex semantic inputs. Modern convolutional neural networks trained on large scale visual data are shown to form similar feature hierarchies [2, 14]. The benefits of transfer learning are generally believed to come from reusing the pre-trained feature hierarchy. This is especially useful when the downstream tasks are too small or not diverse enough to learn good feature representations. However, this intuition cannot explain why in many successful applications of transfer learning, the target domain could be visually very dissimilar to the source domain. To characterize the role of feature reuse, we use a source (pre-train) domain containing natural images (ImageNet), and a few target (downstream) domains with decreasing visual similarities from natural images:

- Semi-transparent represents pretrained model while solid line represents model with random initialization while the numbers on the top of the left fig represents the relative accuracy drop (check formula below in the image description).
- The most intriguing thing to me is that even when the features are very messy, the pretrained models are highly accurate on the training data, though the overfitting is also at the same scale.

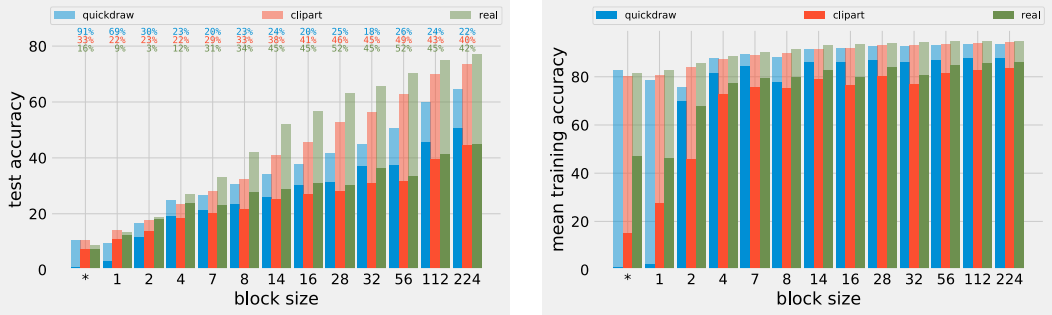


Figure 3: Effects on final performance (left: test accuracy) and optimization speed (right: average training accuracy over 100 finetune epochs) when the input images of the downstream tasks are block-shuffled. The x-axis shows the block sizes for shuffling. Block size '*' is similar to block size '1' except that pixel shuffling operates *across* all RGB channels. For each bar in the plots, the semi-transparent and solid bar correspond to initializing with pre-trained weights and random weights, respectively. The numbers on the top of the left pane show the relative accuracy drop: $100(A_{P\text{-}T} - A_{RI\text{-}T})/A_{P\text{-}T}\%$, where $A_{P\text{-}T}$ and $A_{RI\text{-}T}$ are test accuracy of models trained from pre-trained and random weights, respectively.

DomainNet `real`, DomainNet `clipart`, CheXpert and DomainNet `quickdraw` (see Figure 1). Comparing RI-T to P-T in Figure 2, we observe largest performance boost on the `real` domain, which contains natural images that share similar visual features with ImageNet. This confirms the intuition that feature reuse plays an important role in transfer learning. On the other hand, even for the most distant target domains such as CheXpert and `quickdraw`, we still observe performance boosts from transfer learning. Moreover, apart from the final performance, the optimization for P-T also converges much faster than RI-T in all cases. This suggests additional benefits of pre-trained weights that are not directly coming from feature reuse.

Nothing new here. This is all we already know

To further verify the hypothesis, we create a series of modified downstream tasks which are increasingly distant from normal visual domains. In particular, we partition the image of the downstream tasks into equal sized blocks and shuffle the blocks randomly. The shuffling disrupts high level visual features in those images but keeps the low level statistics about the pixel values intact. The extreme case of block size 224×224 means no shuffling; in the other extreme case, all the pixels in the image are shuffled, making any of the learned visual features in pre-training completely useless. See Figure 1 for examples of block-shuffled images. Note that even for 1×1 blocks, all the RGB channels are moved around *together*. So we created a special case where pixels in each channel move independently and could move to other channels. We then compare RI-T with P-T on those tasks[2]. The impacts on final performance and optimization speed with different block sizes are shown in Figure 3.

We observe that 1) The final performance drops for both RI-T and P-T as the block size decreases, indicating the increasing difficulty of the tasks. 2) The relative accuracy difference, measured as $(A_{P\text{-}T} - A_{RI\text{-}T})/A_{P\text{-}T}\%$, decreases with decreasing block size on both `real` and `clipart`, showing consistency with the intuition that decreasing feature reuse leads to diminishing benefits. 3) On the other hand, on `quickdraw`, the relative accuracy difference does not show a decreasing pattern as in the other two domains. This indicates that for `quickdraw`, where the input images are visually dissimilar to natural images, some other factors from the pre-trained weights are helping the downstream tasks. 4) The optimization speed on P-T is relatively stable, while on RI-T drops drastically with smaller block sizes. This suggests that benefits of transferred weights on optimization speed is independent from feature reuse.

We conclude that feature reuse plays a very important role in transfer learning, especially when the down-

[2]We disable data augmentation (random crop and left-right flip) during finetuning because they no longer make sense for block-shuffled images and make optimization very unstable.

- CNNs are designed for taking into account the visual relationships. Locality, equivariance, and weight sharing are the fundamentals of those nets.
- The more and more you mess with the spatial structures, the poorer the performance is seen on the test set. The question is how a pretrained model is able to provide a good accuracy on the training data? Is it just over-fitting or is it more than that? Think about it.
- If the target domain is very different from the pretrained dataset, feature reuse won't play much of a big role, but a pretrained network will still converge faster than a randomly initialised model because of the "well-behaved" weights.

- But why are the weights well behaved? This is because of the "extra" low-level statistics, information about an image that can be captured and represented as a histogram/distribution.

stream task shares similar visual features with the pre-training domain. But there are other factors at play: in these experiments we change the size of the shuffled blocks all the way to 1 and even try shuffling the channels of the input, therefore, the only object that is preserved here is the set of all pixel values which can be treated as a histogram/distribution. We refer to those information as *low-level statistics*, to suggest that they are void of visual/semantic structural information. The low-level statistics lead to significant benefits of transfer learning, especially on optimization speed.

3.2 Opening up the model

Investigating mistakes In order to understand the difference between different models, we go beyond the accuracy values and look into the mistakes the models make on different samples of the data. We look into the *common mistakes* where both models classify the data point incorrectly and the *uncommon mistakes* (disagreements between models) where only one of them classifies the data point incorrectly and the other one does it correctly. We first compare the ratio of common and uncommon mistakes between two P-Ts, a P-T and a RI-T and two RI-Ts. We note a considerable number of uncommon mistakes between P-T and RI-T models while two P-Ts have strictly fewer uncommon mistakes. This trend is true for both CheXpert and DomainNet target domains.

We visualize the common and uncommon mistakes of each model on DomainNet, and observe that the data samples where P-T is incorrect and RI-T is correct mostly include ambiguous examples; whereas the data samples where P-T is correct, RI-T is incorrect include a lot of easy samples too. This complies with the intuition that since P-T has stronger prior, it harder to adapt to the target domain. Moreover, when we repeat the experiment for another instance of P-T, the mistaken examples between two instances of P-T are very similar, see Appendix B.2 for visualizations of such data points. This may suggest that two P-T's are more similar in the feature space compared to two RI-T's and P-T vs RI-T. We investigate this idea further by looking into similarity of the two networks in the feature space.

Feature Similarity We use the *centered kernel alignment* (CKA) [27] as a measure of similarity between two output features in a layer of a network architecture given two instances of such network. CKA [27] is the latest work on estimating feature similarity with superior performance over earlier works. The results are shown in Table 1. We observe that two instances of P-T are highly similar across different layers. This is also the case when we look into similarity of P-T and P. However, between P-T and RI-T instance or two RI-T instances, the similarity is very low. Note that the feature similarity is much stronger in the penultimate layer than any earlier layers both between P-T and RI-T instance and two RI-T instances, however, still an order of magnitude smaller than similarity between two P-T layers.

These experiments show that the initialization point, whether pre-trained or random, drastically impacts feature similarity, and although both networks are showing high accuracy, they are not that similar in the feature space. This emphasizes on role of feature reuse and that two P-T are reusing the same features.

Table 1: Feature similarity for different layers of ResNet-50, target domain CheXpert

models/layer	conv1	layer 1	layer 2	layer 3	layer 4
P-T & P	0.6225	0.4592	0.2896	0.1877	0.0453
P-T & P-T	0.6710	0.8230	0.6052	0.4089	0.1628
P-T & RI-T	0.0036	0.0011	0.0022	0.0003	0.0808
RI-T & RI-T	0.0016	0.0088	0.0004	0.0004	0.0424

Two networks with same architecture but with random initialization make sense for comparison, but how on this earth two "copies" of the same pretrained model make sense for comparison?
They are expected to be very similar in the feature space

Table 2: Features ℓ_2 distance between two P-T and two RI-T for different target domains

domain/model	2 P-T	2 RI-T	P-T & P	RI-T & P
CheXpert	200.12	255.34	237.31	598.19
clipart	178.07	822.43	157.19	811.87
quickdraw	218.52	776.76	195.44	785.22
real	193.45	815.08	164.83	796.80

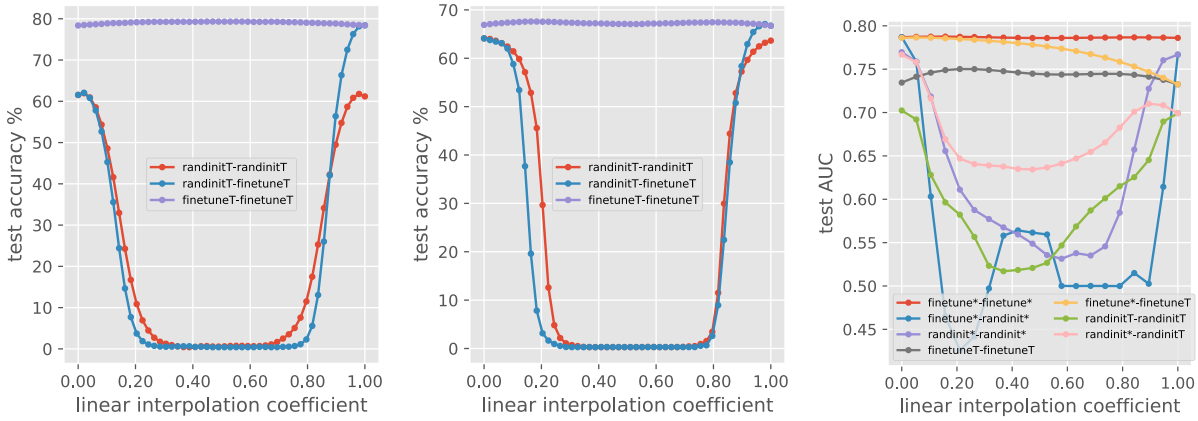


Figure 4: Performance barrier between different solutions. The left and middle panes show performance barrier measured by test accuracy on DomainNet `real` and `quickdraw`, respectively. The right pane shows the performance barrier measured by test AUC on CheXpert. In the legend, 'randinit*' and 'randinitT' means the best and final checkpoint in a RI-T training trajectory. Similarly, 'finetune*' and 'finetuneT' are for P-T. Since there is no overfitting from overtraining on DomainNet, we only show results for final checkpoints.

Distance in parameter space In addition to feature similarity, we look into the distance between two models in the parameter space. More specifically, we measure the ℓ_2 distance between 2 P-Ts and 2 RI-Ts, both per module and for the entire network. Interestingly, RI-Ts are farther from each other compared to two P-Ts, (see Table 2) and this trend can be seen in individual modules too (see Figure 10, 11, Table 8 in the Appendix for more comparisons). Moreover the distance between modules increases as we move towards higher layers in the network.

3.3 Performance barriers and basins in the loss landscape

A commonly used criterion for better generalization performance is the flatness of the basin of the loss landscape near the final solution. In a flat basin, the weights could be locally perturbed without hurting the performance, while in a narrow basin, moving away from the minimizer would quickly hit a *barrier*, indicated by a sudden increase in the loss.

To explore the loss landscape of P-T and RI-T, we use the following procedure to identify potential performance barriers. Let Θ and $\tilde{\Theta}$ be all the weights from two different checkpoints. We evaluate a series of models along the *linear* interpolation of the two weights: $\{\Theta_\lambda = (1-\lambda)\Theta + \lambda\tilde{\Theta} : \lambda \in [0,1]\}$. It has been observed in the literature that any two minimizers of a deep network can be connected via a *non-linear* low-loss path [11, 9, 10]. In contrast, due to the non-linear and compositional structure of neural networks, the *linear*

combination of the weights of two good performing models does not necessarily define a well behaved model, thus performance barriers are generally expected along the linear interpolation path. However, in the case when the two solutions belong to the same flat basin of the loss landscape, the linear interpolation remains in the basin. As a result, a performance barrier is absent. Moreover, interpolating two random solutions from the same basin could generally produce solutions closer to the center of the basin, which potentially have better generalization performance than the end points.

Like SWA?

The word "basin" is often used loosely in the literature to refer to areas in the parameter space where the loss function has relatively low values. Since prior work showed that non-linear low-loss path could be found to connect any pair of solutions, we focus on convex hull and linear interpolation in order to avoid trivial connectivity results. In particular, we require that for most points on the basin, their convex combination is on the basin as well. This extra constraint would allow us to have multiple basins that may or may not be connected though a low-loss (nonlinear) path. We formalize this notion as follows:

Constraint

Definition 3.1. *Given a loss function $\ell : \mathbb{R}^n \to \mathbb{R}^+$ and a closed convex set $S \subset \mathbb{R}^n$, we say that S is a (ϵ, δ)-basin for ℓ if and only if S has all following properties:*

1. *Let U_S be the uniform distribution over set S and $\mu_{S,\ell}$ be the expected value of the loss ℓ on samples generated from U_S. Then,*

$$\mathbb{E}_{\mathbf{w}\sim U_S}[|\ell(\mathbf{w}) - \mu_{S,\ell}|] \leq \epsilon \tag{1}$$

2. *For any two points $w_1, w_2 \in S$, let $f(w_1, w_2) = w_1 + \tilde{\alpha}(w_2 - w_1)$, where $\tilde{\alpha} = \max\{\alpha | w_1 + \alpha(w_2 - w_1) \in S\}$. Then,*

$$\mathbb{E}_{\mathbf{w_1},\mathbf{w_2}\sim U_S,\nu\sim\mathcal{N}(0,(\delta^2/n)I_n)}[\ell(f(\mathbf{w_1},\mathbf{w_2}) + \nu) - \mu_{S,\ell}] \geq 2\epsilon \tag{2}$$

3. *Let $\kappa(\mathbf{w_1}, \mathbf{w_2}, \nu) = f(\mathbf{w_1}, \mathbf{w_2}) + \frac{\nu}{\|f(\mathbf{w_1},\mathbf{w_2}) - \mathbf{w_1}\|_2}(f(\mathbf{w_1}, \mathbf{w_2}) - \mathbf{w_1})$. Then,*

$$\mathbb{E}_{\mathbf{w_1},\mathbf{w_2}\sim U_S,\nu\sim\mathcal{N}(0,\delta^2)}[\ell(\kappa(\mathbf{w_1},\mathbf{w_2},|\nu|)) - \mu_{S,\ell}] \geq 2\epsilon \tag{3}$$

Explanation

Based on the above definition, there are three requirements for a convex set to be a basin. The first requirement is that for most points on the basin, their loss should be close to the expected value of the loss in the basin. This notion is very similar to requiring the loss to have low variance for points on the basin[3]. The last two requirements ensure that the loss of points in the vicinity of the basin is higher than the expected loss on the basin. In particular, the second requirement does that by adding Gaussian noise to the points in the basin and requiring the loss to be higher than the expected loss in the basin. The third requirement does something similar along the subspaces spanned by extrapolating the points in the basin. That is, if one exits the basin by extrapolating two points on the basin, the loss should increase.

In Figure 4 we show interpolation results on DomainNet `real`, `quickdraw`, and CheXpert. Generally, we observe no performance barrier between the P-T solutions from two random runs, which suggests that the pre-trained weights guide the optimization to a flat basin of the loss landscape. Moreover, there are models along the linear interpolation that performs slightly better than the two end points, which is again consistent with our intuition. On the other hand, barriers are clearly observed between the solutions from two RI-T runs (even if we use the same random initialization values). On CheXpert, because the models start to overfit after certain training epochs (see Figure 2), we examine both the best and final checkpoints along the training trajectory, and both show similar phenomena. Interestingly, the interpolation between the best and the final checkpoints (from the same training trajectory) also shows a (small) barrier for RI-T (but not for P-T).

[3]Here the term that captures the difference has power one as opposed to variance where the power is two.

Starting with the two P-T solutions, we extrapolated beyond their connecting intervals to find the basin boundary, and calculated the parameters according to Definition 3.1. We found that each pair of P-T solutions live in a $(0.0038, 49.14)$-basin, $(0.0054, 98.28)$-basin and $(0.0034, 49.14)$-basin for `real`, `clipart` and `quickdraw`, respectively. Their $\mu_{S,\ell}$ values are 0.2111, 0.2152 and 0.3294, respectively, where ℓ measures the test error rate. On the other hand, pairs of RI-T solutions do not live in the same (ϵ, δ)-basin for reasonable ϵ thresholds.

We have done a variety of additional experiments on analysing the performance barriers for different domains, cross-domains, extrapolation and training on combined domains that can be found in the Appendix.

3.4 Module Criticality

It has been observed that different layers of the network show different robustness to perturbation of their weight values [45]. Zhang et al. [45] did the following experiment: consider a trained network, take one of the modules and rewind its value back to its initial value while keeping the weight value of all other modules fixed at trained values. They noted that for some modules, which they called *critical*, the performance of the model drops significantly after rewinding, while for others the performance is not impacted. Chatterji et al. [4] investigated this further and formulated it by a notion of module criticality. Module criticality is a measure that can be calculated for each module and captures how critical each module is. It is defined in [4] by looking at width of the valleys that connect the final and initial value of weight matrix of a module, while keeping all the other modules at their final trained value.

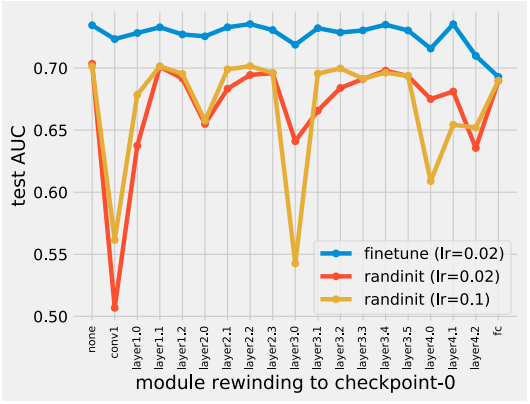


Figure 5: Module criticality measured on CheXpert.

Definition 3.2 (Module Criticality [4]). *Given an $\epsilon > 0$ and network f_Θ, we define the module criticality for module i as follows:*

$$\mu_{i,\epsilon}(f_\Theta) = \min_{0 \le \alpha_i, \sigma_i \le 1} \left\{ \frac{\alpha_i^2 \left\| \theta_i^F - \theta_i^0 \right\|_{\text{Fr}}^2}{\sigma_i^2} : \mathbb{E}_{u \sim \mathcal{N}(0, \sigma_i^2)} [\mathcal{L}_S(f_{\theta_i^\alpha + u, \Theta_{-i}^F})] \le \epsilon \right\}, \tag{4}$$

where $\theta_i^\alpha = (1-\alpha)\theta_i^0 + \alpha\theta_i^F, \alpha \in [0,1]$ is a point on convex combination path between the final and initial value of the weight matrix θ_i and we add Gaussian perturbation $u \sim \mathcal{N}(0, \sigma_i^2)$ to each point.

Chatterji et al. [4] showed that module criticality captures the role of the module in the generalization performance of the whole architecture and can be used as a measure of capacity of the module and predict the generalization performance. In this paper, we extend the definition of module criticality by looking at both *direct path* that linearly connect the initial and final value of the module and the *optimization path* generated by an optimizer from initialization to the final solution (checkpoints during training). We also look into the final value of a weight matrix in addition to its optimal value during training as the start point of the path for investigating criticality, i.e., instead of θ_i^F, we investigate θ_i^{opt} which is the checkpoint where the network has the best validation accuracy during training. Moreover, we ensure that the noise is proportional to the Frobenius norm of weight matrix. Similar to [4], we define the network criticality as the sum of the module criticality over modules of the network. Note that we can readily prove the same relationship between this definition to generalization performance of the network as the one in [4] since the proof does not depend on the start point of the path.

- Module criticality is a method to capture how critical a module is for generalisation and better performance.
- To measure the criticality of a module, you fix the weights of other modules and then tries to go back from final weights to initial weight values.
- If you can go all the way back to initial values without significant drop in performance (less than some threshold epsilon), then that module isn't critical

- As you move back, and the loss starts to increase, that means the module is critical in that case.
- Non-critical modules generally have wider valleys.

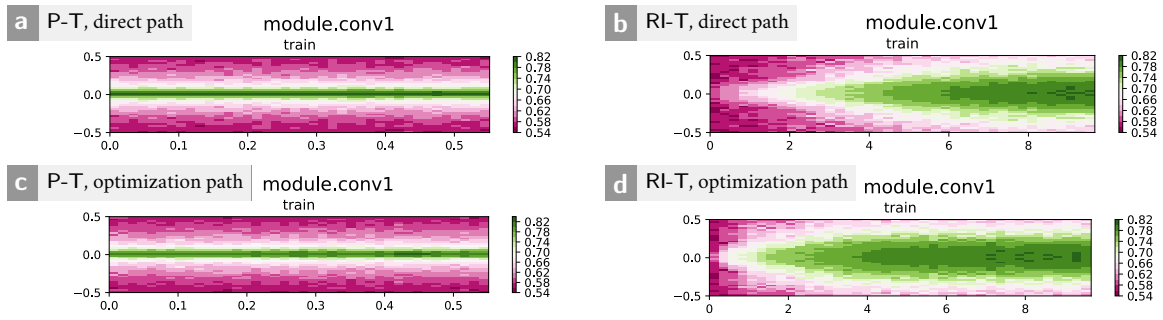


Figure 6: Module Criticality plots for Conv1 module. x-axis shows the distance between initial and optimal θ, where $x = 0$ maps to initial value of θ. y-axis shows the variance of Gaussian noise added to θ. The four subplots refer to the four paths one can use to measure criticality. All of which provide good insight into this phenomenon. Heat map is used so that the colors reflect the value of the measure under consideration.

In Figure 5 we analyze criticality of different modules in the same way as [45]. We note a similar pattern as observed in the supervised case. The only difference is that the 'FC' layer becomes critical for P-T model, which is expected. Next, we investigate crititcality of different modules with our extended definition along with original definition. We note that both optimization and direct paths provide interesting insights into the criticality of the modules. We find that the optimal value of weight is a better starting point for this analysis compared to its final value. Figure 6 shows this analysis for 'Conv1' module, which as shown in Figure 5 is a critical module. In this Figure we only look at the plots when measuring performance on training data. As shown in Figure 30 (Appendix B.9), we can see the same trend when looking at the test performance or generalization gap.

As we move from the input towards the output, we see tighter valleys, i.e., modules become more critical. This is in agreement with observation of [44, 36] that lower layers are in charge of more general features while higher layers have features that are more specialized for the target domain. Moreover, For modules of the RI-T model, we notice more sensitivity and a transition point in the path between optimal and initial point, where the valley becomes tighter and wider as we move away from this point. Whether this is related to the module moving to another basin is unclear.

3.5 Which pre-trained checkpoint is most useful for transfer learning?

We compare the benefits of transfer learning by initializing the pre-trained weights from different checkpoints on the pre-training optimization path. Figure 7 shows the final performance and optimization speed when finetuning from different pre-training checkpoints. Overall, the benefits from pre-training increase as the checkpoint index increases. Closer inspection reveals the following observations: 1) in pre-training, big performance boosts are observed at epoch 30 and 60 where the learning rate decays. However, initializing from checkpoints 29, 30, 30 (and similarly 59, 60, 61) does not show significantly different impact. On the other hand, especially for final performance of `real` and `clipart`, significant improvements are observed when we start from the checkpoints where the pre-training performance has been plateauing (i.e. checkpoint 29 and 59). This shows that the pre-training performance is not always a faithful indicator of the effectiveness of the pre-trained weights for transfer learning. 2) `quickdraw` sees much smaller benefit on final performance from pre-training and quickly plateaus at checkpoint 10, while `real` and `clipart` continuously see noticeable performance improvements until checkpoint 60. On the other hand, all three tasks get significant benefits on optimization speed improvements as the checkpoint index increases. 3) The optimization speedups start to plateau at checkpoint 10, while (for

Nothing new

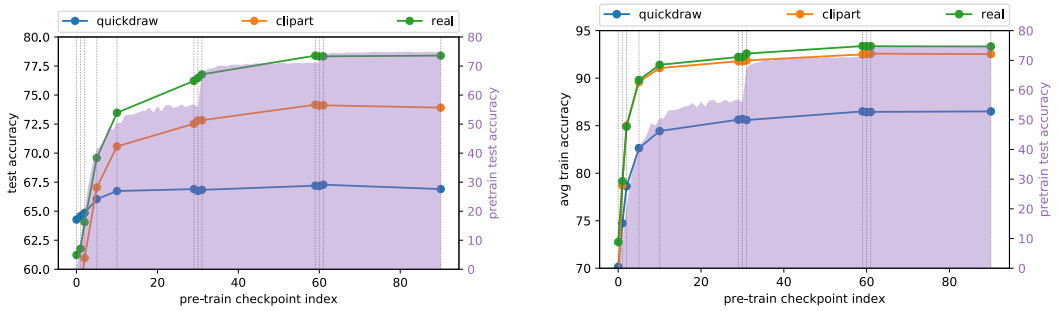


Figure 7: Comparing the final performance (left: test accuracy after finetuning) and optimization speed (right: average training accuracy over 100 finetune epochs) of transfer learning from different pre-training checkpoints. The purple shaded area indicates the top-1 accuracy of the ImageNet pre-training task at each of the checkpoints.

`real` and `clipart`) the final performance boosts continue to increase.

In summary, we observe independence between the improvements on optimization speed and final performance. Moreover, this is in line with the loss landscape observations in Section 3.3. Earlier checkpoints in pre-training are out of basin of the converged model and at some point during training we enter the basin. This also explains the plateau of performance after some checkpoints. Therefore, we can start from earlier checkpoints in pre-training.

4 Related work

Recent work has looked into whether transfer learning is always successful [18, 25, 31, 20, 12]. For example, Kornblith et al. [25] illustrates that pretrained features may be less general than previously thought and He et al. [18] show that transfer (even between similar tasks) does not necessarily result in performance improvements. Kolesnikov et al. [24] propose a heuristic for setting the hyperparameters for transfer. Yosinski et al. [44] show that in visual domain features from lower layers are more general and as we move more towards higher layers the features become more specific. Raghu et al. [35] evaluated the role of feature reusing in meta-learning. Directly related to our work is [36], where they investigate transfer learning from pre-trained ImageNet model to medical domain and note the role of model size on transfer performance, as well as the role of feature independent aspects such as weight scaling. In this paper, we go beyond their angles and propose an extensively complementary analysis. We dissect the role of different modules from various angles. Moreover, we provide insights into what is being transferred. We also extend the understanding of role of feature reuse and other parameters at play for successful transfer.

Transfer learning is also a key components in recent breakthroughs in natural language processing, as pre-trained representations from task-agnostic transformer language models turned out to work extremely well on various downstream tasks [8, 34, 43, 28, 3], therefore extensively analyzed [19, 41]. Transfer learning in text domains is charasteristically different from transfer learning in visual domains due to the nature of the data, utility of untrained representations, pre-training techniques and the amount of distribution shifts between pre-training and downstream tasks. In this paper, we focus on studying transfer learning in visual domain.

The flatness of the basin of the loss landscape near the final solution is studied as an indicator of the generalization performance of neural network models [23, 22]. It is found that a *nonlinear* connecting path could be found between any pair of basins corresponding to different solutions [11, 9, 10]. In this paper, we study *linear* path connectivity between solutions to investigate the connectivity in the parameter space.

5 Conclusion and future work

In this paper we shed some light on what is being transferred in transfer learning and which parts of the network are at play. We investigated the role of feature reuse through shuffling the blocks of input and showed that when trained from pre-trained weights initialization, the network stays in the same basin of the solution, features are similar and models are close in the ℓ_2 distance in parameter space. We also confirmed that lower layers are in charge of more general features. Our findings on basin in the loss landscape can be used to improve ensemble methods. Our observation of low-level data statistics improving training speed could lead to better network initialization methods. Using these findings to improve transfer learning is of interest for future work. More specifically, we plan to look into initialization with minimum information from pre-trained model while staying in the same basin and whether this improves performance. For example, one can use top singular values and directions for each module for initialization and investigate if this suffices for good transfer, or ensure initialization at the same basin but adding randomization to enhance diversity and improve generalization. Another interesting direction is the implications of these findings for parallel training and optimization. Since we are staying in the same basin, we can take model average without disturbing the performance.

Acknowledgements

We would like to thank Samy Bengio and Maithra Raghu for valuable conversations.

References

[1] P. L. Bartlett, D. J. Foster, and M. J. Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, pages 6240–6249, 2017.

[2] D. Bau, B. Zhou, A. Khosla, A. Oliva, and A. Torralba. Network dissection: Quantifying interpretability of deep visual representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6541–6549, 2017.

[3] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[4] N. S. Chatterji, B. Neyshabur, and H. Sedghi. The intriguing role of module criticality in the generalization of deep networks. *ICLR*, 2020.

[5] J. Darrell, J. Long, and E. Shelhamer. Fully convolutional networks for semantic segmentation. *IEEE T PATTERN ANAL*, 39(4), 2014.

[6] J. De Fauw, J. R. Ledsam, B. Romera-Paredes, S. Nikolov, N. Tomasev, S. Blackwell, H. Askham, X. Glorot, B. O'Donoghue, D. Visentin, et al. Clinically applicable deep learning for diagnosis and referral in retinal disease. *Nature medicine*, 24(9):1342, 2018.

[7] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[9] F. Draxler, K. Veschgini, M. Salmhofer, and F. Hamprecht. Essentially no barriers in neural network energy landscape. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1309–1318, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[10] S. Fort and S. Jastrzebski. Large scale structure of neural network loss landscapes. In *Advances in Neural Information Processing Systems*, pages 6709–6717, 2019.

[11] T. Garipov, P. Izmailov, D. Podoprikhin, D. P. Vetrov, and A. G. Wilson. Loss surfaces, mode connectivity, and fast ensembling of dnns. In *Advances in Neural Information Processing Systems*, pages 8789–8798, 2018.

[12] R. Geirhos, P. Rubisch, C. Michaelis, M. Bethge, F. A. Wichmann, and W. Brendel. Imagenet-trained cnns are biased towards texture; increasing shape bias improves accuracy and robustness. In *ICLR*, 2019.

[13] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[15] C. G. Gross. Genealogy of the "grandmother cell". *The Neuroscientist*, 8(5):512–518, 2002.

[16] K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016.

[17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[18] K. He, R. Girshick, and P. Dollár. Rethinking imagenet pre-training. *arXiv preprint arXiv:1811.08883*, 2018.

[19] D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained transformers improve out-of-distribution robustness. *arXiv preprint arXiv:2004.06100*, 2020.

[20] M. Huh, P. Agrawal, and A. A. Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016.

[21] J. Irvin, P. Rajpurkar, M. Ko, Y. Yu, S. Ciurea-Ilcus, C. Chute, H. Marklund, B. Haghgoo, R. Ball, K. Shpanskaya, et al. Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 590–597, 2019.

[22] Y. Jiang, B. Neyshabur, H. Mobahi, D. Krishnan, and S. Bengio. Fantastic generalization measures and where to find them. *ICLR*, 2019.

[23] N. S. Keskar, D. Mudigere, J. Nocedal, M. Smelyanskiy, and P. T. P. Tang. On large-batch training for deep learning: Generalization gap and sharp minima. *arXiv preprint arXiv:1609.04836*, 2016.

[24] A. Kolesnikov, L. Beyer, X. Zhai, J. Puigcerver, J. Yung, S. Gelly, and N. Houlsby. Large scale learning of general visual representations for transfer. *arXiv preprint arXiv:1912.11370*, 2019.

[25] S. Kornblith, J. Shlens, and Q. V. Le. Do better imagenet models transfer better? *arXiv preprint arXiv:1805.08974*, 2018.

[26] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019.

[27] S. Kornblith, M. Norouzi, H. Lee, and G. Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97, pages 3519–3529. PMLR, 09–15 Jun 2019.

[28] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[29] D. Mahajan, R. Girshick, V. Ramanathan, K. He, M. Paluri, Y. Li, A. Bharambe, and L. van der Maaten. Exploring the limits of weakly supervised pretraining. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 181–196, 2018.

[30] B. Neyshabur, S. Bhojanapalli, and N. Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. *ICLR*, 2018.

[31] J. Ngiam, D. Peng, V. Vasudevan, S. Kornblith, Q. V. Le, and R. Pang. Domain adaptive transfer learning with specialist models. *arXiv preprint arXiv:1811.07056*, 2018.

[32] K. Pearson. Vii. note on regression and inheritance in the case of two parents. *proceedings of the royal society of London*, 58(347-352):240–242, 1895.

[33] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, and B. Wang. Moment matching for multi-source domain adaptation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1406–1415, 2019.

[34] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019.

[35] A. Raghu, M. Raghu, S. Bengio, and O. Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[36] M. Raghu, C. Zhang, J. Kleinberg, and S. Bengio. Transfusion: Understanding transfer learning for medical imaging. In *Advances in Neural Information Processing Systems*, pages 3342–3352, 2019.

[37] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. *CoRR*, arXiv:1711.05225, 2017.

[38] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[39] H. Sedghi, V. Gupta, and P. Long. The singular values of convolutional layers. In *Proceedings of the International Conference on Learning Representations*, 2019.

[40] C. Sun, A. Shrivastava, S. Singh, and A. Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[41] A. Tamkin, T. Singh, D. Giovanardi, and N. Goodman. Investigating transferability in pretrained language models. *arXiv preprint arXiv:2004.14975*, 2020.

[42] X. Wang, Y. Peng, L. Lu, Z. Lu, M. Bagheri, and R. M. Summers. Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3462–3471. IEEE, 2017.

[43] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in neural information processing systems*, pages 5753–5763, 2019.

[44] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson. How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328, 2014.

[45] C. Zhang, S. Bengio, and Y. Singer. Are all layers created equal? *arXiv preprint arXiv:1902.01996*, 2019.

[46] H. Zhang, Y. N. Dauphin, and T. Ma. Fixup initialization: Residual learning without normalization. *arXiv preprint arXiv:1901.09321*, 2019.

Appendix

Table of Contents

Appendix A: Experiment Setup

We use ImageNet [7] pre-training for its prevalence in the community and consider CheXpert [21] and three sets from DomainNet [33] as downstream transfer learning tasks. CheXpert is a medical imaging dataset which consists of chest x-ray images. We resized the x-ray images to to 224×224, and set up the learning task to diagnose 5 different thoracic pathologies: atelectasis, cardiomegaly, consolidation, edema and pleural effusion. DomainNet [33] is a dataset of common objects in six different domain. All domains include 345 categories (classes) of objects such as Bracelet, plane, bird and cello . The domains include `clipart`: collection of clipart images; `real`: photos and real world images; `sketch`: sketches of specific objects; `infograph`: infographic images with specific object; `painting` artistic depictions of objects in the form of paintings and `quickdraw`: drawings of the worldwide players of game "Quick Draw!"[4]. In our experiments we use three domains: `real`, `clipart` and `quickdraw`.

The CheXpert dataset default split contains a training set of 200k images and a tiny validation set that contains only 200 studies. The drastic size difference is because the training set is constructed using an algorithmic labeler based on the free text radiology reports while the validation set is manually labeled by board-certified radiologists. To avoid high variance in the studies due to tiny dataset size and label distribution shift, we do not use the default partition. Instead, we sample two separate sets of 50k examples from the full

[4] `https://quickdraw.withgoogle.com/data`

training set and use them as the train and test set, respectively. Different domains in DomainNet contain different number of training examples, ranging from 50k to 170k. To keep the setting consistent with CheXpert, We sample 50k subsets from training and test sets for each of the domains.

For all our training and transfer experiments we use ResNet-50 [16] with fixup initialization [46] which eliminate batch normalization layers from the ResNet architecture. We initialize the final linear classifier layer with uniform random values instead of using zeros from the fixup initialization.

We run each CheXpert training jobs with two NVidia V100 GPUs, using batch size 256. We use the SGD optimizer with momentum 0.9, weight decay 0.0001, and constant learning rate scheduling. We also tried piece-wise constant learning rate scheduling. However, we found that RI-T struggles to continue learning as learning rate decays. We train P-T for 200 epochs and RI-T for 400 epochs, long enough for both training scenarios to converge to the final (overfitted) solutions. It takes ~ 90 seconds to train one epoch. We also run full evaluation on both the training and test sets each epoch, which takes ~ 40 seconds each.

We run each DomainNet training jobs with single NVidia V100 GPU, using batch size 32. We use the SGD optimizer with momentum 0.9, weight decay 0.0001, and piecewise constant learning rate scheduling that decays the learning rate by a factor of 10 at epoch 30, 60, and 90, respectively. We run the training for 100 epochs. In each epoch, training takes around 2 minutes and 45 seconds, and evaluating on both the training and test set takes around 70 seconds each.

Appendix B: Additional Figures and Tables

B.1 Discussions of learning curves

Figure 2 shows the learning curves for P-T, RI-T models with different learning rates. In particular, we show base learning rate 0.1 and 0.02 for both cases on CheXpert, DomainNet `real`, `clipart` and `quickdraw`, respectively.

We observe that P-T generally prefer smaller learning rate, while RI-T generally benefit more from larger learning rate. With large learning rate, P-T failed to converge on CheXpert, and significantly under-performed the smaller learning rate counterpart on DomainNet `quickdraw`. On DomainNet `real` and `clipart`, the gap is smaller but smaller learning rate training is still better. On the other hand, with larger learning rate, RI-T significantly outperform the smaller learning rate counterpart on all the three DomainNet datasets. On CheXpert, although the optimal and final performance are similar for both small and large learning rate, the large learning version converges faster.

On all the four datasets, P-T outperforms RI-T, in both optimization speed and test performance. Note that we subsample all the four dataset to $50,000$ training examples of 224×224 images, yet severe overfitting is observed only on CheXpert. This suggests that the issue of overfitting is not only governed by the problem size, but also by other more complicated factors such as the nature of input images. Intuitively, the chest X-ray images from CheXpert are less diverse than the images from DomainNet.

B.2 Common and uncommon mistakes

In order to look into common and uncommon mistakes, we compare two models at a time. We look into all combinations, i.e., compare RI-T, P-T, two instances of RI-T and two instances of P-T. Tables 3, 4, 5 show this analysis for CheXpert and Table 6 shows the analysis for `clipart`. For CheXpert we do a per class analysis first. Since CheXpert looks into five different diseases, we have five binary classification tasks. In each classification setting, we look into accuracy of each model. g1, g2 refer to the number of data samples where only the first model is classifying correctly and only the second model is classifying correctly, respectively. We also look into

Table 3: Common and uncommon mistakes between RI-T, P-T, CheXpert

	P-T acc	RI-T acc	g1	g2	Common mistakes	r1	r2
Class 1	0.8830	0.8734	645	832	3597	0.1878	0.1520
Class 2	0.7490	0.7296	2361	2318	4047	0.3641	0.3684
Class 3	0.9256	0.9095	71	207	3009	0.0643	0.02305
Class 4	0.6811	0.6678	2250	2460	8835	0.2177	0.2029
Class 5	0.7634	0.7374	3446	2160	4200	0.3396	0.4506

Table 4: Common and uncommon mistakes between two instances of RI-T, CheXpert

	P-T acc	RI-T acc	g1	g2	Common mistakes	r1	r2
Class 1	0.8734	0.8654	704	773	3469	0.1822	0.1687
Class 2	0.7292	0.7399	3139	1499	4909	0.2339	0.3900
Class 3	0.9095	0.9170	195	102	2978	0.0331	0.0614
Class 4	0.6678	0.6574	2070	2588	8497	0.2334	0.1958
Class 5	0.7374	0.7326	2001	3136	4510	0.4101	0.3073

the number of common mistakes. r1, r2 refer to ratio of uncommon to all mistakes for the first and second model respectively.

Another interesting point about DomainNet is that classes are not balanced. Therefore, we investigate the correlation of P-T and RI-T accuracy with class size. The results are shown in Figure 8 for `clipart`. Other target domains from DomainNet show similar trends. We also compute the Pearson correlation coefficient [32] between per class accuracy's and class sizes for these models and they are P-T $(0.36983, 1.26e-12)$, RI-T $(0.32880, 3.84e-10)$. Note that overall top-1 accuracy of P-T, RI-T is $74.32, 57.91$ respectively. In summary, P-T has higher accuracy overall, higher accuracy per class and it's per-class accuracy is more correlated with class size compared to RI-T.

B.3 Feature similarity and different distances

Table 7 shows feature similarity using CKA [26] for output of different layers of ResNet-50 when the target domain is `clipart`. In Section 3.2 in the main text we showed a similar table for target domain CheXpert. We observe a similar trend when we calculate these numbers for `quickdraw` and `real` target domain as well.

Figure 10 depicts ℓ_2 distance between modules of two instances of P-T and two instances of RI-T for both

Table 5: Common and uncommon mistakes between two instances of P-T, CheXpert

	P-T acc	RI-T acc	g1	g2	Common mistakes	r1	r2
Class 1	0.8830	0.8828	456	697	3732	0.1573	0.1088
Class 2	0.7490	0.7611	2375	1369	4996	0.2150	0.3222
Class 3	0.9256	0.9208	63	139	3077	0.0432	0.0200
Class 4	0.6811	0.6825	2030	1879	9416	0.1663	0.1773
Class 5	0.76344	0.76088	1687	2398	3962	0.3770	0.2986

Table 6: Common and uncommon mistakes between P-T, RI-T, clipart

	g1	g2	Common mistakes	r1	r2
P-T, RI-T	521	2940	3263	0.1376	0.4739
RI-T, RI-T	787	844	5359	0.1280	0.1360
P-T, P-T	619	584	3165	0.1635	0.1558

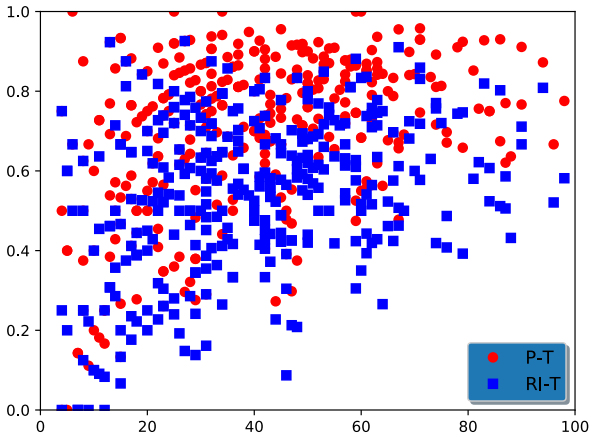
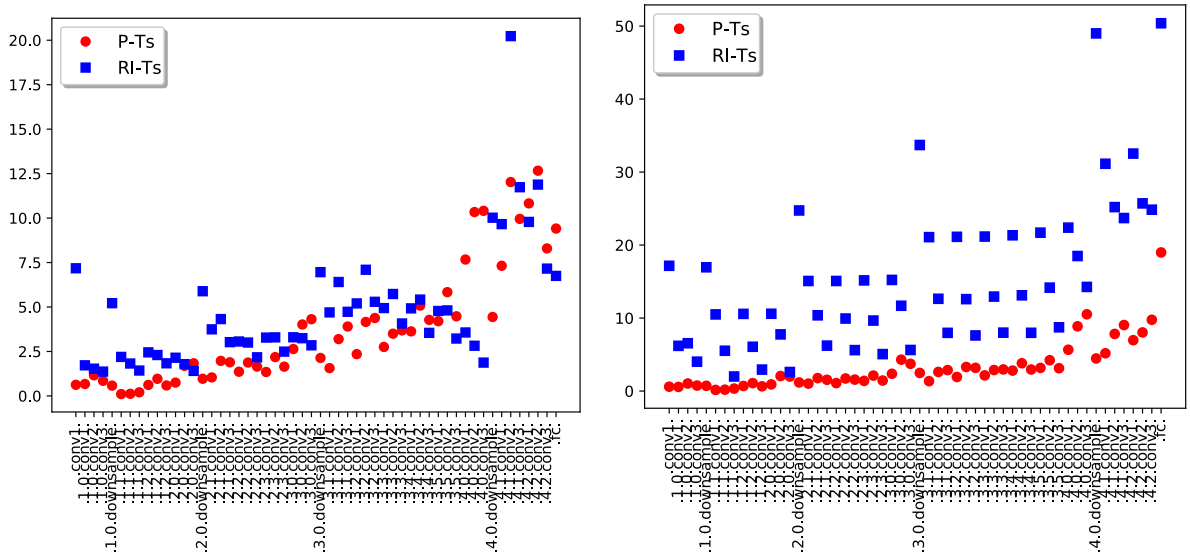


Figure 8: Accuracy of P-T and RI-T vs class size for clipart

Table 7: Feature similarity for different layers of ResNet-50, target domain clipart

models/layer	conv1	layer 1	layer 2	layer 3	layer 4
P-T & P	0.3328	0.2456	0.1877	0.1504	0.5011
P-T & P-T	0.4333	0.1943	0.4297	0.2578	0.1167
P-T & RI-T	0.0151	0.0028	0.0013	0.008	0.0014
RI-T & RI-T	0.0033	0.0032	0.0088	0.0033	0.0012

Figure 9: Uncommon Mistakes RI-T and P-T, top row shows samples of the images RI-T classifies incorrectly while P-T classifies them correctly. bottom row shows samples of the images P-T classifies incorrectly while RI-T classifies them correctly. Classes from left to right are: barn, apple, backpack, angel.



(a) CheXpert (b) clipart

Figure 10: Feature ℓ_2 distance per module

Table 8: Distance to initialization between P-T and RI-T for different target domains

domain/model	P-T	RI-T
CheXpert	4984	4174
clipart	5668	23249
quickdraw	7501	24713
real	5796	24394

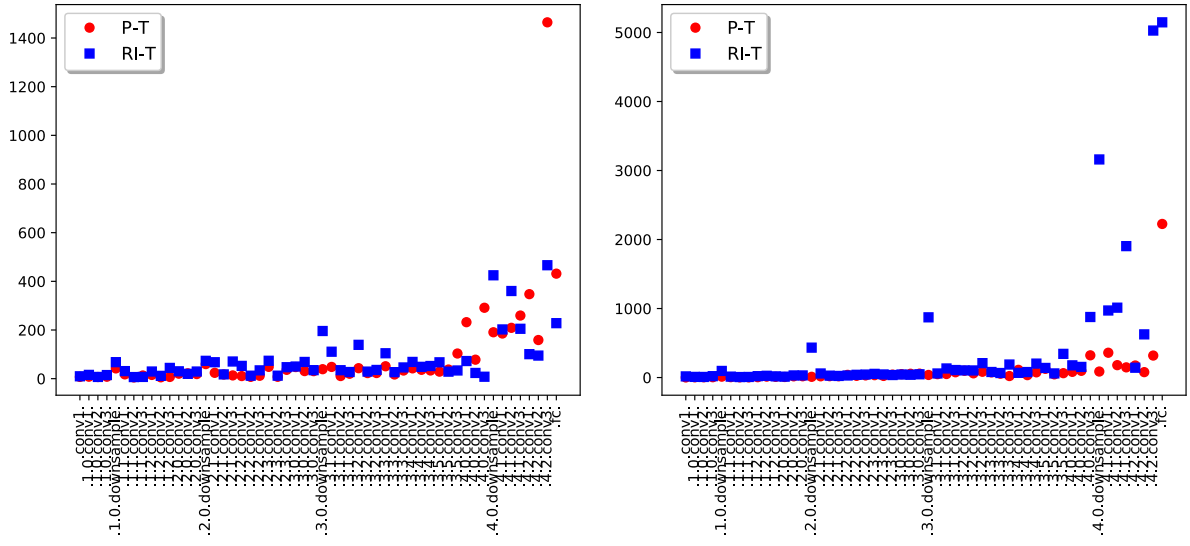
CheXpert and clipart target domain. Table 2 shows the overall distance between the two networks. We note that P-T's are closer in ℓ_2 parameter domain compared to RI-Ts. Drawing this plot for real, quickdraw leads to the same conclusion.

We also looked into distance to initialization per module, and overall distance to initialization for P-T, RI-T for different target domains in Figure 11 and Table 8.

B.4 Additional plots for performance barriers

Figure 12 shows the performance barrier plots measured with on all the three DomainNet datasets. Figure 13 show results measured with the cross entropy loss. On both plots, we observe performance barrier between two RI-T solutions, but not between two P-T solutions.

Figure 14 and Figure 15 show the performance barrier plots of CheXpert measured by AUC and loss, respectively. The two panes in each of the figures show the two cases where RI-T is trained with learning rate 0.1 and 0.02, respectively. Note that on CheXpert the models overfit afer a certain number of training epochs and the final performances are worse than the optimal performances along the training trajectory. So we show more interpolation pairs than in the case of DomainNet. Those plots are largely consistent with our previous



(a) CheXpert (b) clipart

Figure 11: Distance to Initialization per module

observations. One interesting observation is that while the final performance of P-T is better than RI-T when measured with (test) AUC, the former also has higher (test) loss than the latter.

B.5 Performance barrier experiments with identical initialization for RI-T

In the experiments of comparing the performance barrier interpolating the weights of two RI-T models vs. interpolating the weights of two P-T models, the two P-T models are initialized from *the same* pre-trained weights, while the two RI-T models are initialized from independently sampled (therefore *different*) random weights. In this section, we consider interpolating two RI-T models that are trained from *identical* (random) initial weights. The results are shown in Figure 16 and Figure 17. Comparing with their counterparts in Figure 12 and Figure 13, respectively, we found that the barriers between two RI-T models become slightly smaller when the initial weights are the same. However, significant barriers still exist when comparing with the interpolation between two P-T models.

B.6 Performance barrier plots with extrapolation

In order to estimate the boundary of the basin according to Definition 3.1, we extend the interpolation coefficients from $[0, 1]$ to extrapolation in $[-1, 2]$. The results are shown in Figure 18. We can see that in this 1D subspace, the two P-T solutions are close to the boundary of the basin, while RI-T solutions do not live in the same basin due to the barriers on the interpolating linear paths.

B.7 Cross-domain weight interpolation on DomainNet

Because all the domains in DomainNet have the same target classes, we are able to directly apply a model trained on one domain to a different domain and compute the test performance. Moreover, we can also interpolate the weights between models that are trained on different domains. In particular, we tested the following scenarios:

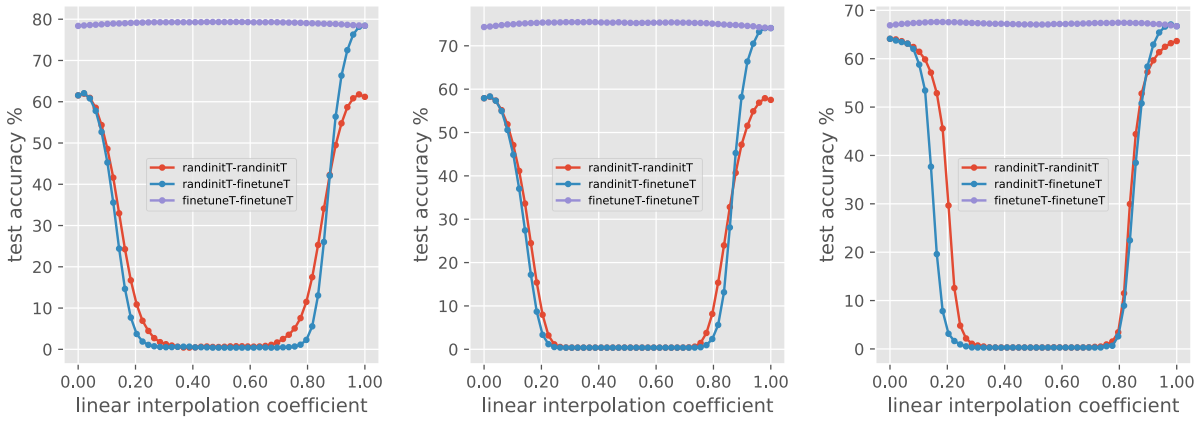


Figure 12: Performance barrier of `real`, `clipart`, `quickdraw`, respectively, measured by test accuracy.

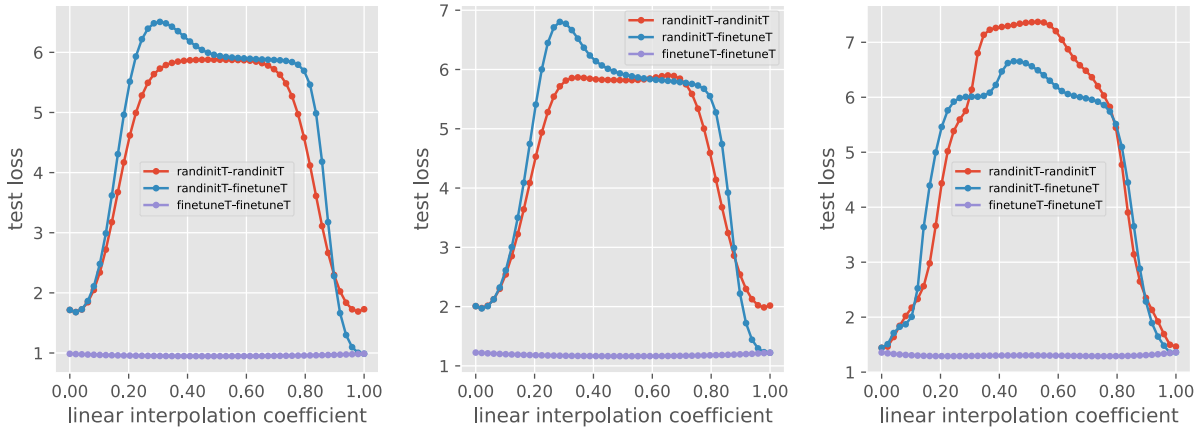


Figure 13: Loss barrier of `real`, `clipart`, `quickdraw`, respectively, measured by the cross entropy loss.

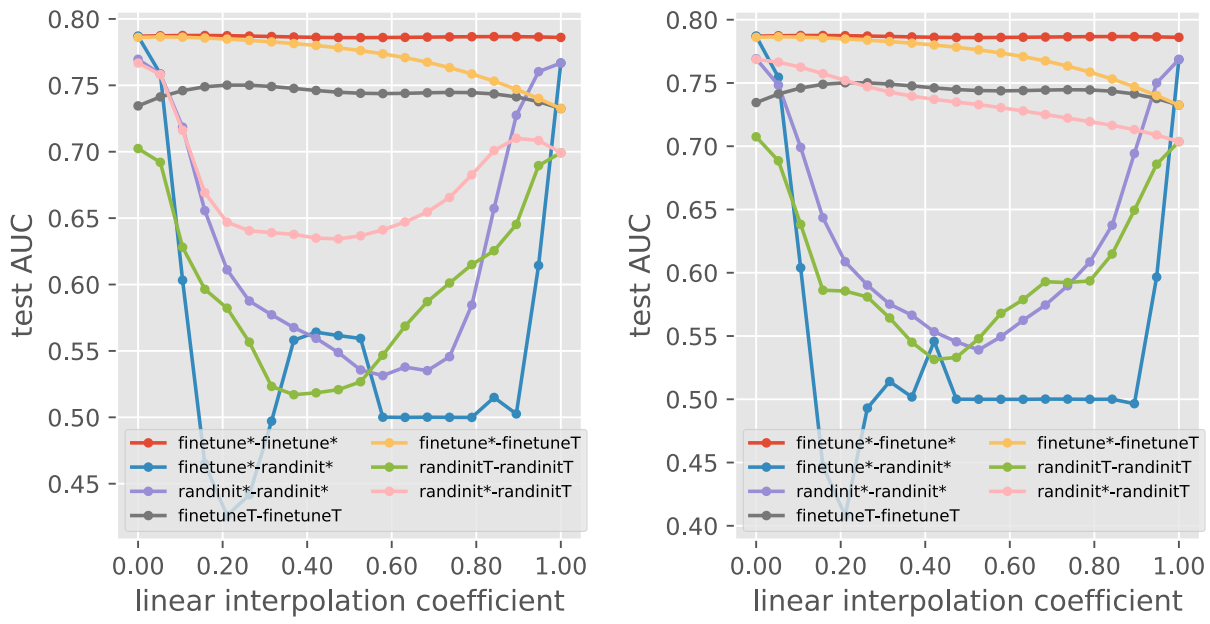


Figure 14: Performance barrier on CheXpert. Left: RI-T is using base learning rate 0.1; Right: RI-T is using base learning rate 0.02.

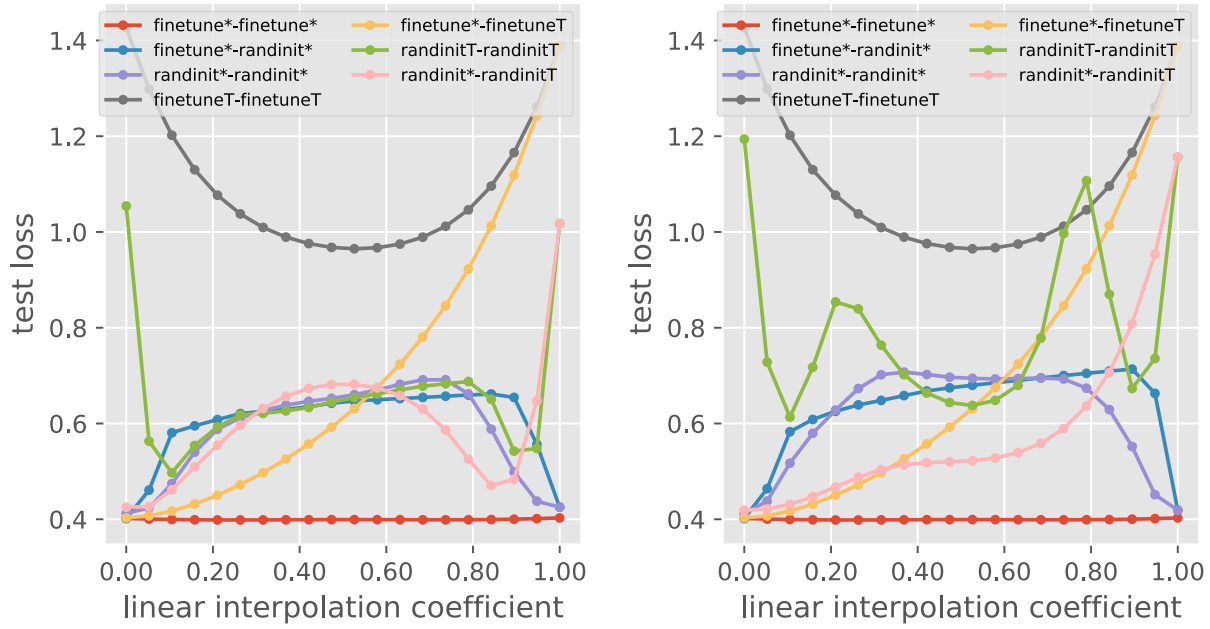


Figure 15: Loss barrier on CheXpert. Left: RI-T is using base learning rate 0.1; Right: RI-T is using base learning rate 0.02.

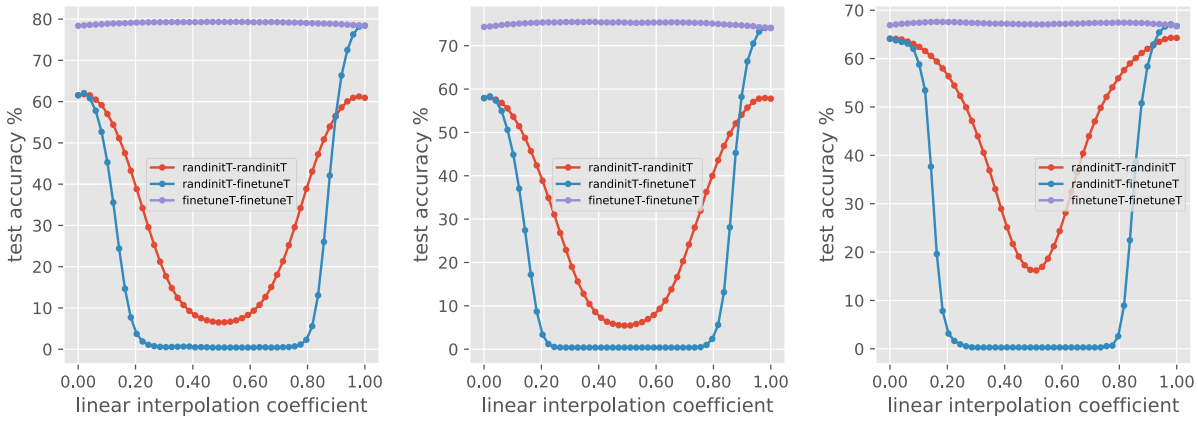


Figure 16: Performance barrier of `real`, `clipart`, `quickdraw`, respectively, measured by test accuracy. Like P-T, the two RI-T models are initialized from *the same* (random) weights. This figure can be compared with Figure 12.

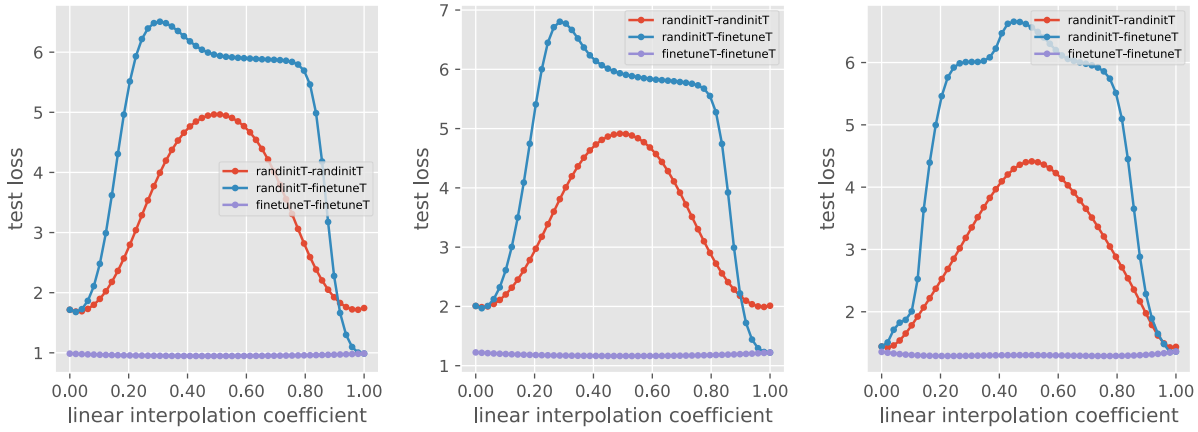


Figure 17: Performance barrier of `real`, `clipart`, `quickdraw`, respectively, measured by cross entropy loss. Like P-T, the two RI-T models are initialized from *the same* (random) weights. This figure can be compared with Figure 13.

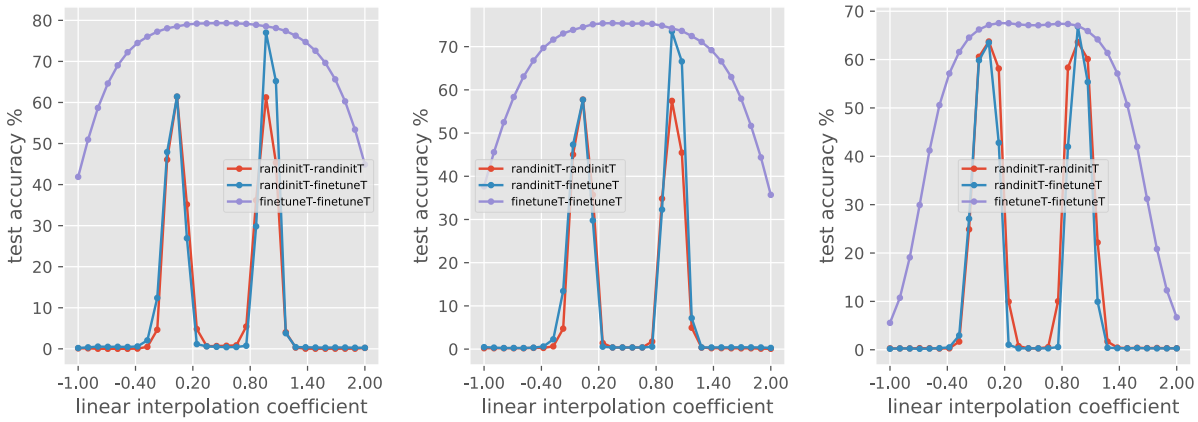


Figure 18: Performance barrier of `real`, `clipart`, `quickdraw`, respectively, measured by test accuracy. The linear combination of weights are extrapolated beyond $[0, 1]$ (to $[-1, 2]$).

Results	Evaluated on	Training data for Model 1	Training data for Model 2
Figure 19	`clipart`	`clipart`	`real`
Figure 20	`clipart`	`quickdraw`	`real`
Figure 21	`real`	`quickdraw`	`clipart`
Figure 22	`real`	`real`	`clipart`
Figure 23	`real`	`real`	`quickdraw`

It is interesting to observe that when directly evaluated on a different domain that the models are trained from, we could still get non-trivial test performance. Moreover, P-T consistently outperforms RI-T even in the cross-domain cases. A more surprising observation is that when interpolating between P-T models, (instead of performance barrier) we observe performance boost in the middle of the interpolation. This suggests that all the trained P-T models on all domains are in one shared basin.

B.8 Cross-domain weight interpolation with training on combined domains

In this section, we investigate interpolation with models that are trained on combined domains. In particular, some models are trained on a dataset formed by the union of the training set from multiple DomainNet domains. We tested the following scenarios:

Results	Evaluated on	Training data for Model 1	Training data for Model 2
Figure 24	`real`	`real+clipart`	`clipart`
Figure 25	`real`	`real+quickdraw`	`quickdraw`
Figure 26	`real`	`real+quickdraw`	`real`
Figure 27	`real`	`clipart+quickdraw`	`clipart`
Figure 28	`clipart`	`real+clipart`	`real`
Figure 29	`clipart`	`real+quickdraw`	`quickdraw`

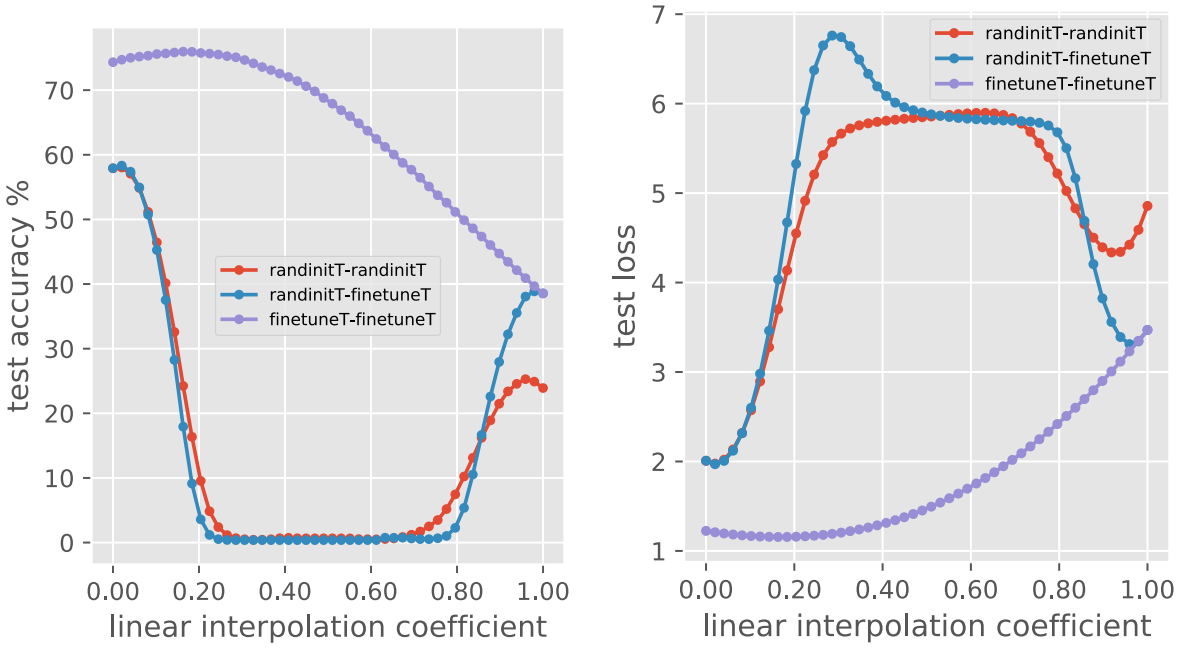


Figure 19: Performance barrier of cross-domain interpolation. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `clipart` domain. The interpolation are between models trained on `clipart` and models trained on `real`.

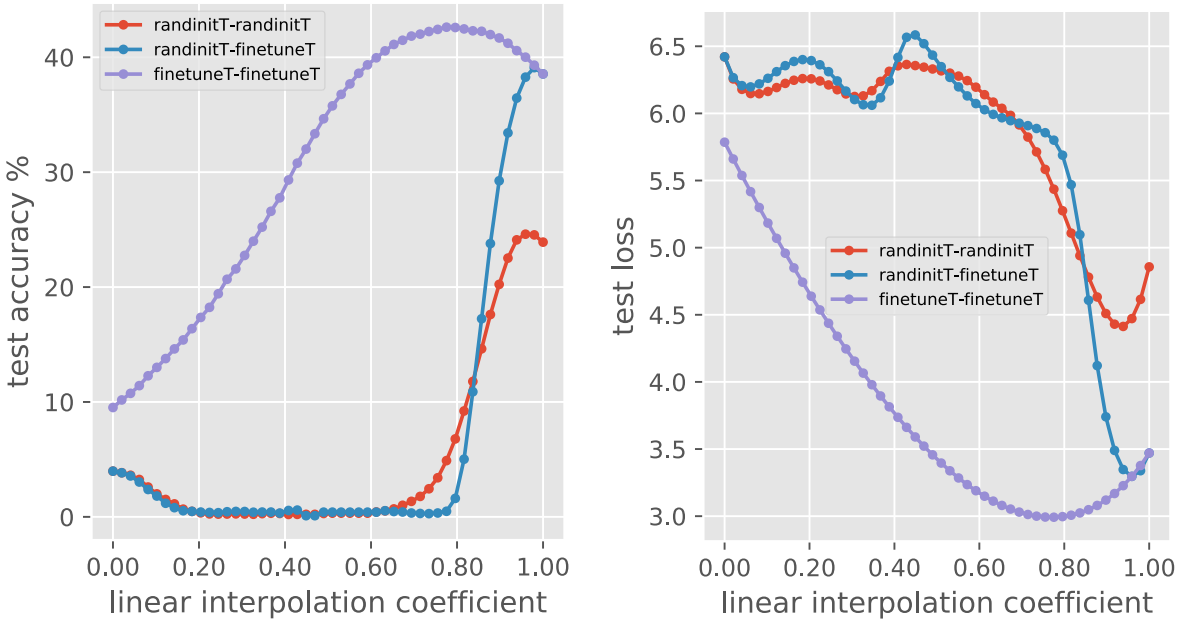


Figure 20: Performance barrier of cross-domain interpolation. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `clipart` domain. The interpolation are between models trained on `quickdraw` and models trained on `real`.

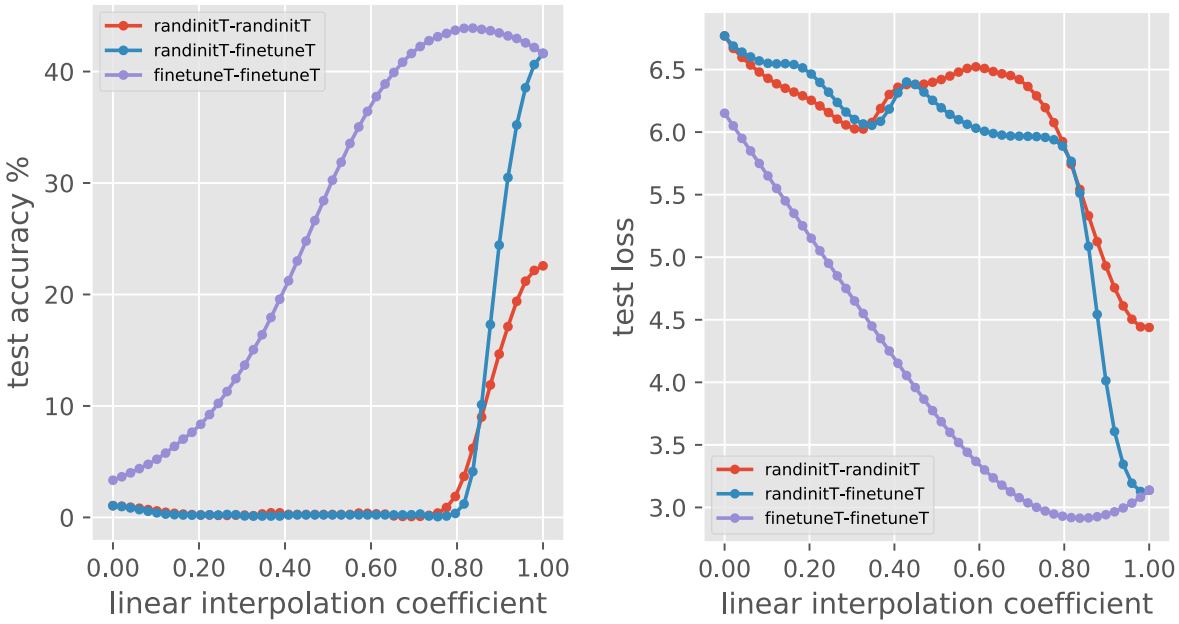


Figure 21: Performance barrier of cross-domain interpolation. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `quickdraw` and models trained on `clipart`.

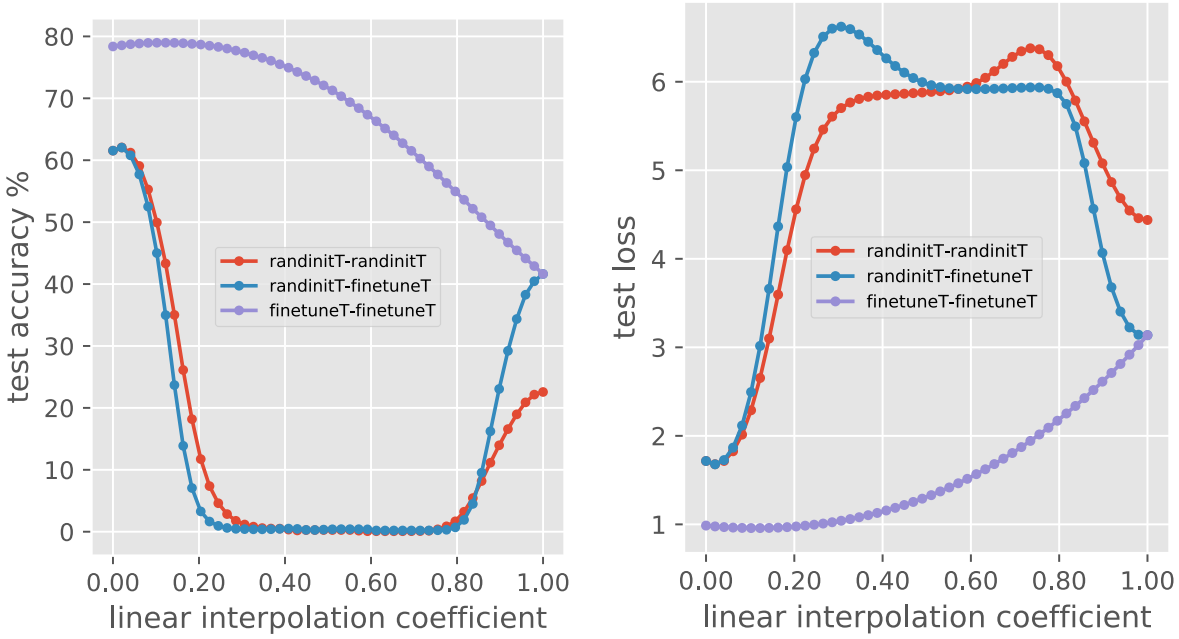


Figure 22: Performance barrier of cross-domain interpolation. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `real` and models trained on `clipart`.

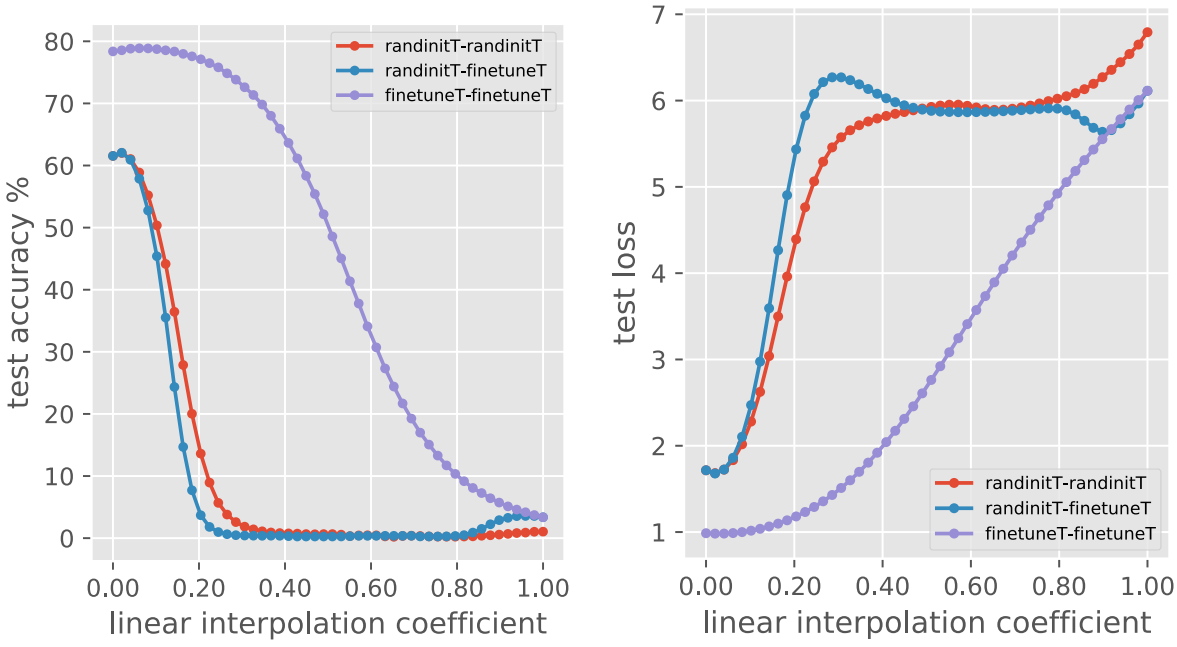


Figure 23: Performance barrier of cross-domain interpolation. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `real` and models trained on `quickdraw`.

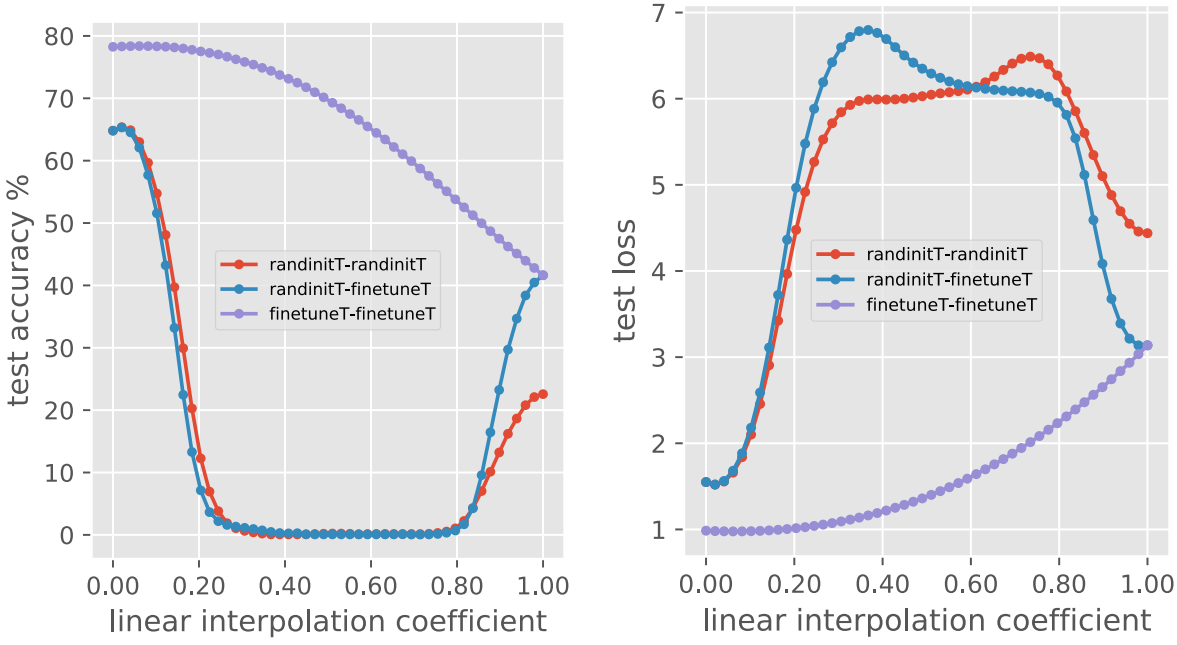


Figure 24: Performance barrier of cross-domain interpolation with training on combined domains. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `real+clipart` and models trained on `clipart`.

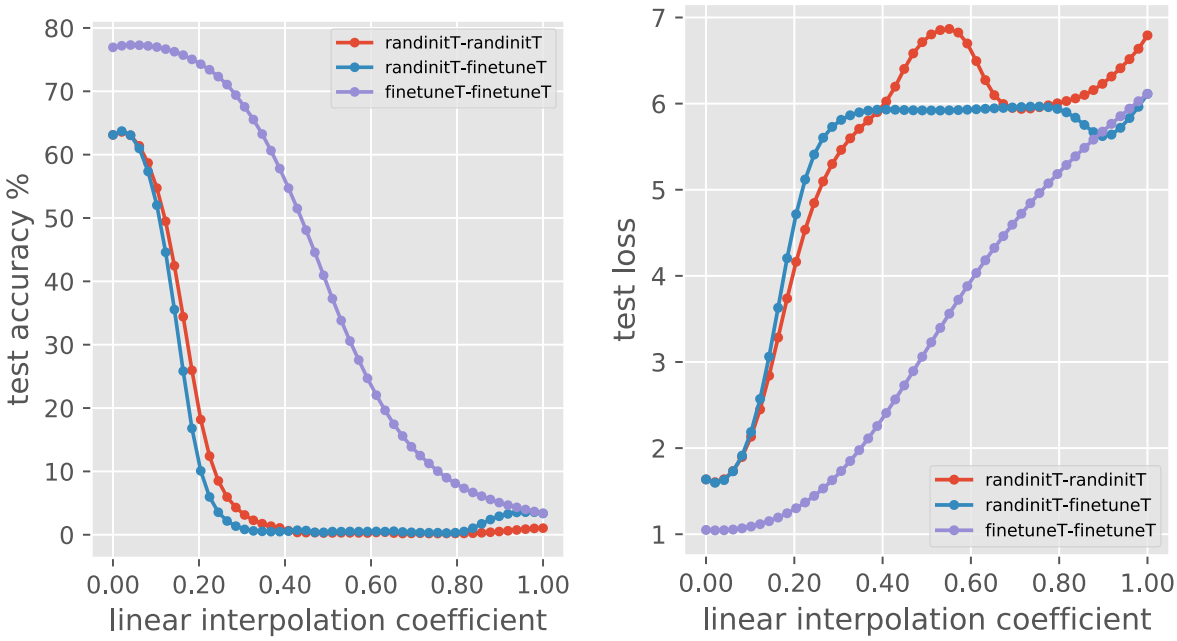


Figure 25: Performance barrier of cross-domain interpolation with training on combined domains. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `real+quickdraw` and models trained on `quickdraw`.

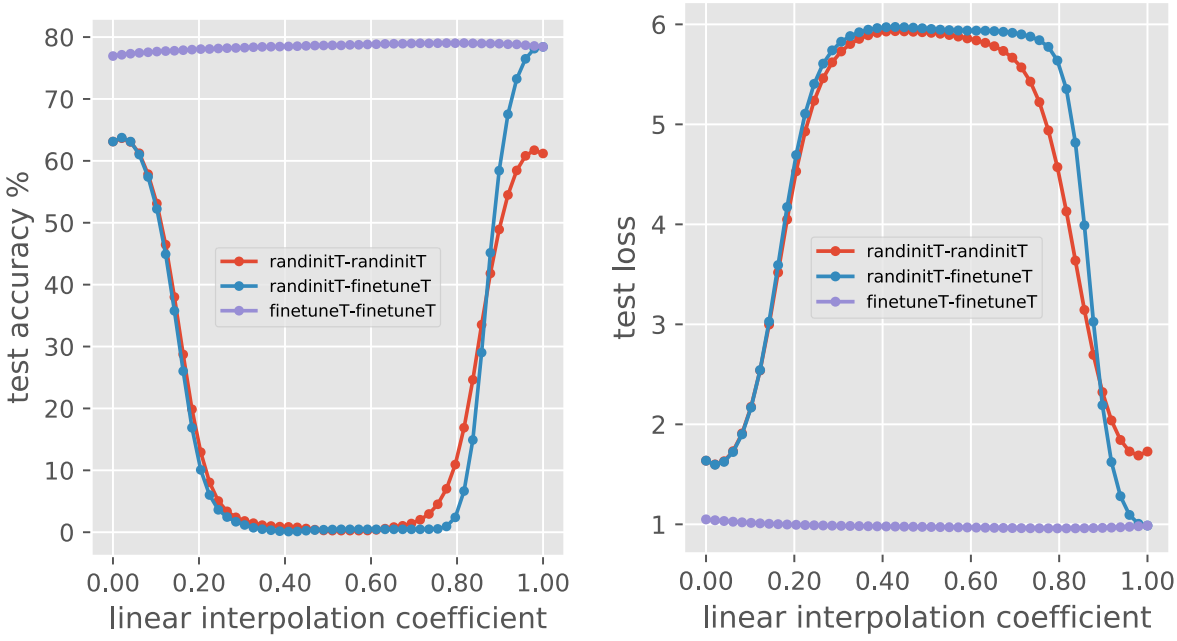


Figure 26: Performance barrier of cross-domain interpolation with training on combined domains. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `real+quickdraw` and models trained on `real`.

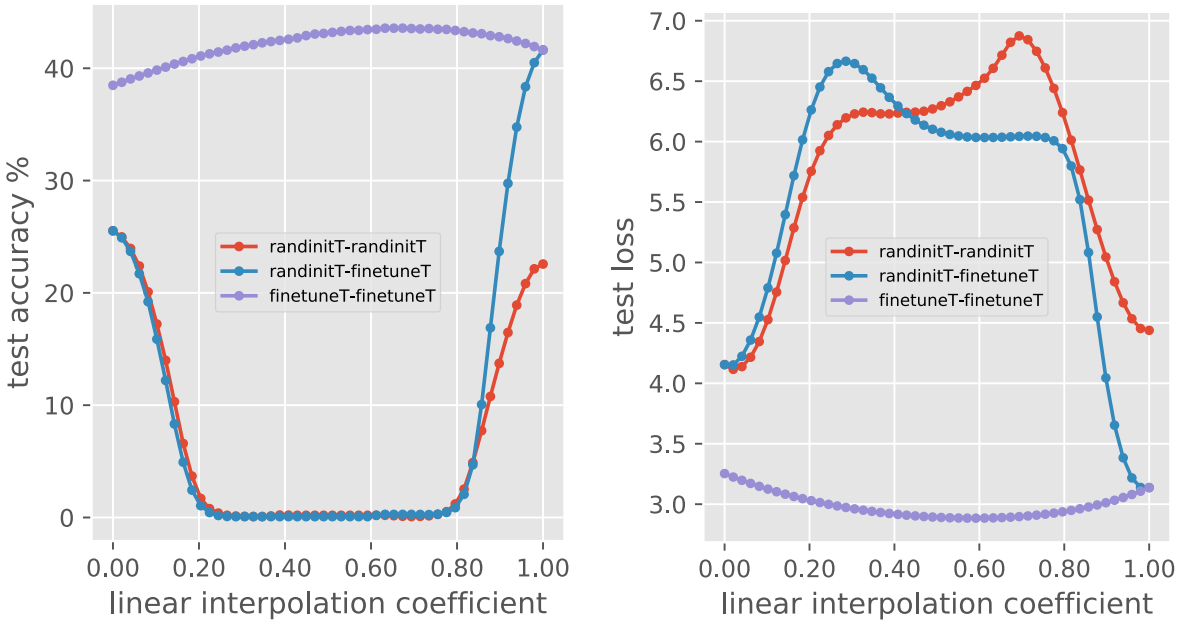


Figure 27: Performance barrier of cross-domain interpolation with training on combined domains. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `real` domain. The interpolation are between models trained on `clipart+quickdraw` and models trained on `clipart`.

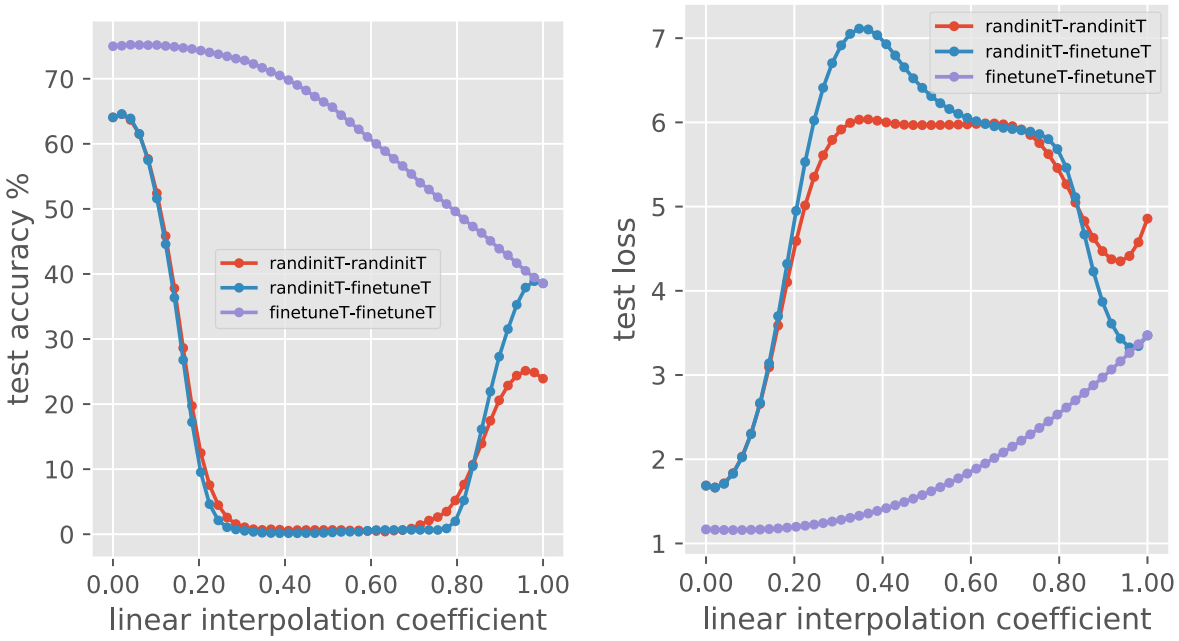


Figure 28: Performance barrier of cross-domain interpolation with training on combined domains. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `clipart` domain. The interpolation are between models trained on `real+clipart` and models trained on `real`.

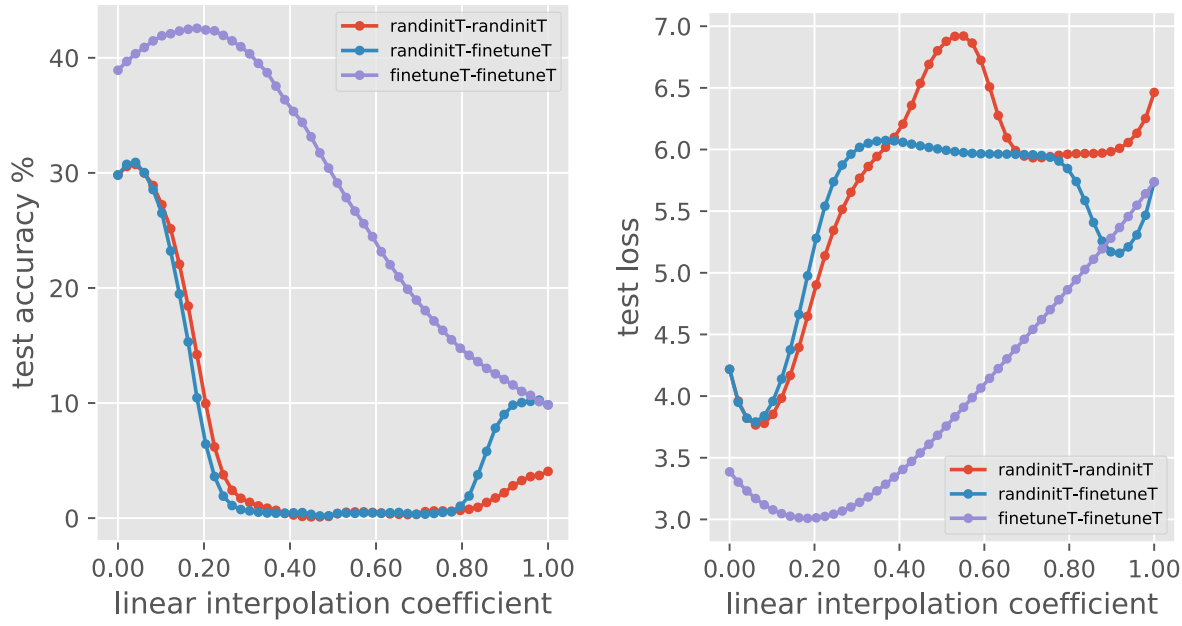


Figure 29: Performance barrier of cross-domain interpolation with training on combined domains. The test accuracy (left) and the cross entropy loss (right) are evaluated on the `clipart` domain. The interpolation are between models trained on `real+quickdraw` and models trained on `quickdraw`.

B.9 Additional criticality plots

Figure 30 in shows the criticality analysis for Conv1 module of the ResNet-50 using training data or test data or generalization gap. As we see, all of them can be used interchangeably for the analysis. The accompanying file 'criticality-plots-chexpert.pdf' includes the figures from main text along with many more such plots for different layers of ResNet-50.

Appendix C: Spectrum of weight matrices

We recover the spectrum of every module using the algorithm derived in [39] and we also look at the spectrum for the whole network in different cases of training.Sedghi et al. [39] proposes an exact and efficient method for finding all singular values corresponidng to the convolution layers with a simple two-lines of NumPy which essentially first takes $2D$-FFT of the kernel and then takes the union of singular values for different blocks of the result to find all singular values. Figure 31a shows the spectrum of the whole network for different models for CheXpert domain. The plots for other domains and for individual modules are shown in the Supplementary material. We note that for individual modules as well as the whole network, RI-T is more concentrated towards zero. In other words, it has a higher density in smaller singular values. This can be seen in Figure 31a, 31b. In order to depict this easier, we sketch the number of singular values smaller than some threshold vs the value of the threshold in Figure 33. Intuitively, among two models that can classify with certain margin, which translates to having low cross-entropy loss, then we can look at concentration of spectrum and concentration towards low values shows less confidence. More mathematically speaking, the confident model requires a lower-rank to get ϵ-approximation of the function and therefore, has lower capacity. Intuitively, distribution around small

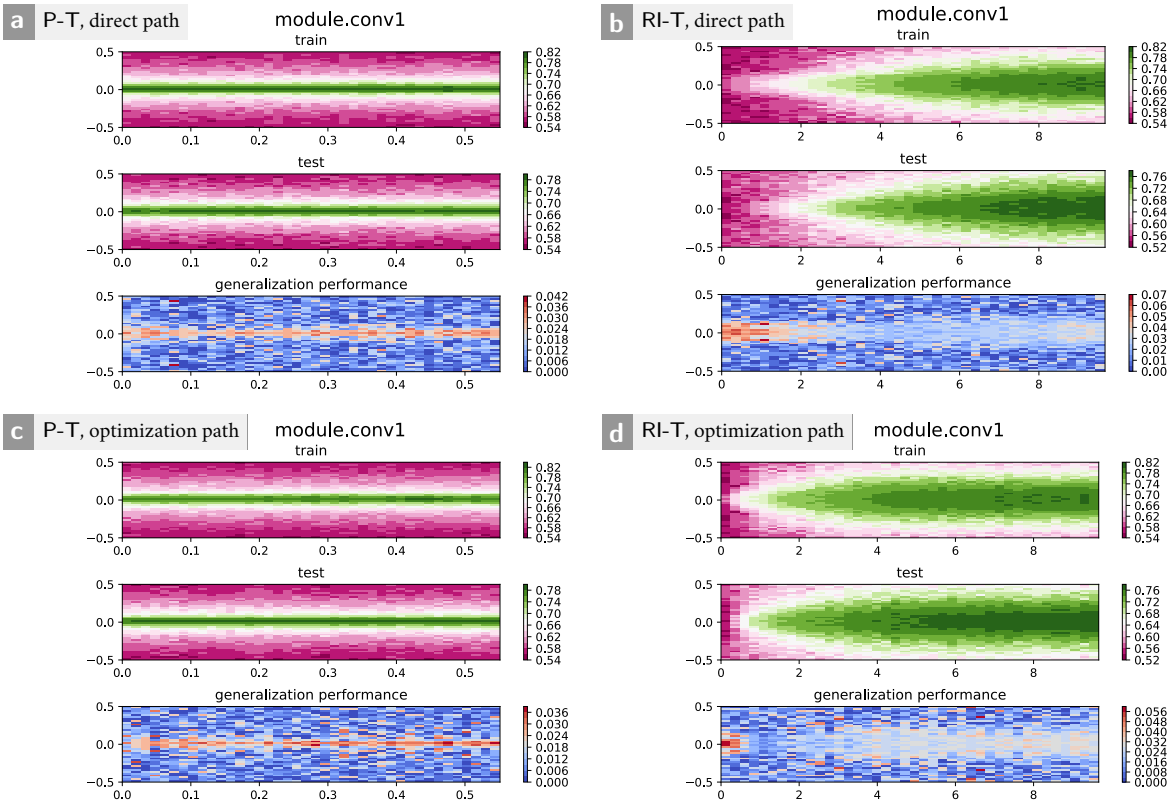
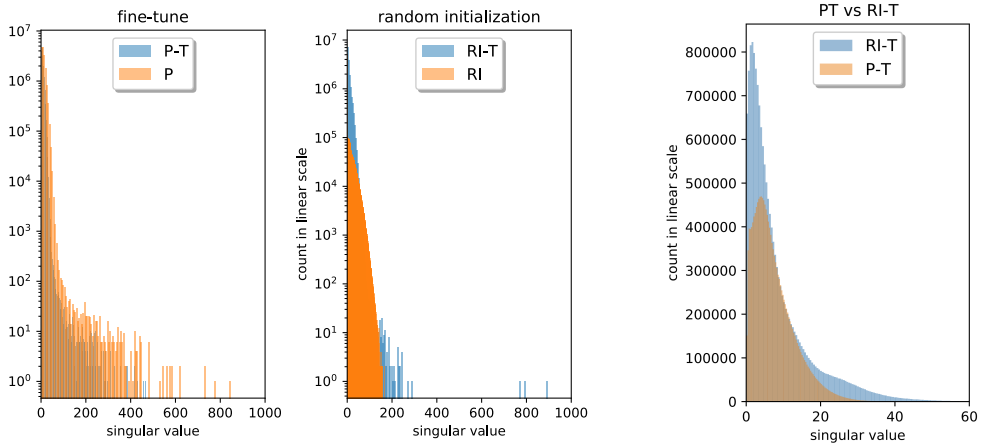


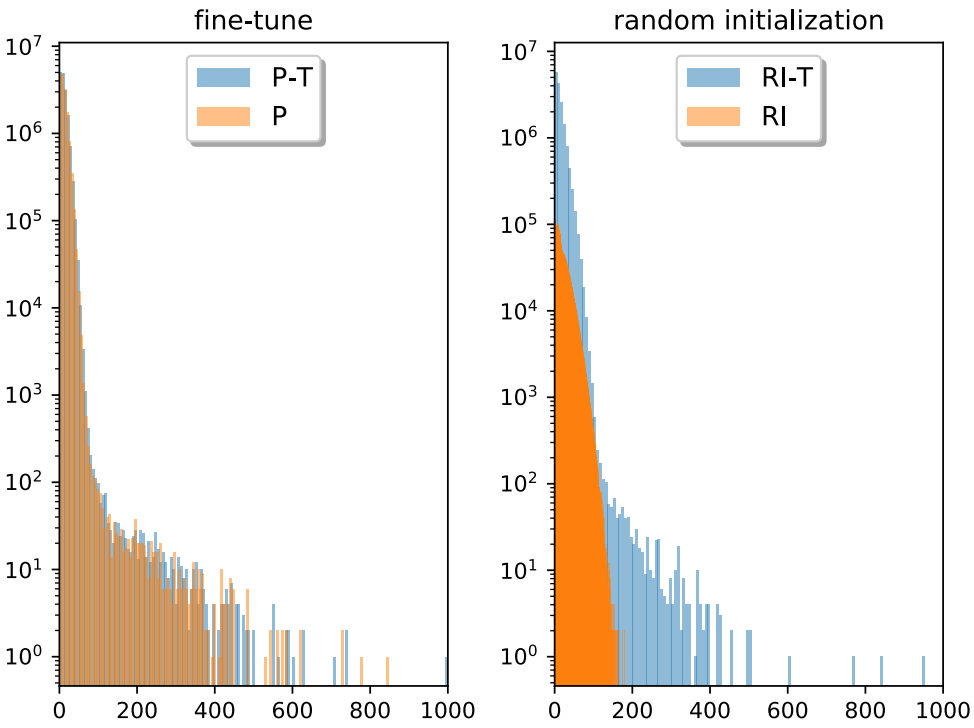
Figure 30: Module Criticality plots for Conv1 module. x-axis shows the distance between initial and optimal θ, where $x = 0$ maps to initial value of θ. y-axis shows the variance of Gaussian noise added to θ. The four subplots refer to the four paths one can use to measure criticality. All of which provide good insight into this phenomenon. Each subplot has 3 rows corresponding to train error, test error and generalization error. Heat map is used so that the colors reflect the value of the measure under consideration.



(a) Histogram of spectrum of the whole model

(b) Lower part of spectrum

Figure 31: Spectrum of the whole network, CheXpert.



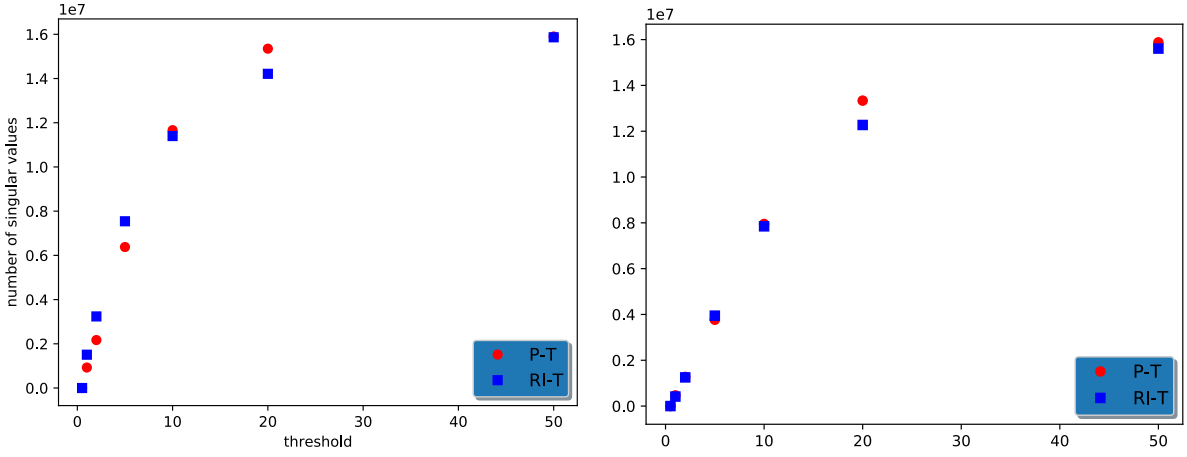
Importance of singular values

Figure 32: Spectrum of the whole model for target domain Clipart

singular values is a hint of model uncertainty. When starting from pre-trained network, the model is pointing strongly into directions that have signals about the data. Given that RI-T does not start with strong signals about the data, it finds other explanations compared to P-T and hence ends up in a different basin of loss landscape, which from a probabilistic perspective is more concentrated towards smaller singular values.

Figure 32 shows the spectrum of the whole network for target domain `clipart`. The accompanying files 'spectrum-plots-chexpert.pdf', 'spectrum-plots-clipart.pdf' in the Supplementary material folder include the spectrum for each module of ResNet-50 as well as the whole spectrum for target domain CheXpert, `clipart` respectively.

There is a vast literature in analyzing generalization performance of DNNs by considering the ratio of Frobenius norm to spectral norm for different layers [1, 30]. They prove an upper bound on generalization error in the form of $O\left(1/\gamma, B, d, \Pi_{i\in l}\|\theta_i\|_2 \sum_{i\in l} \frac{\|\theta_i\|_F}{\|\theta_i\|_2}\right)$ where γ, B, d, l refer to margin, norm of input, dimension of the input, depth of the network and Θ_is refer to module weights and Frobenius and spectral norm are shown with $\|\cdot\|_F, \|\cdot\|_2$. For details, see [30]. The product of spectral norm can be considered as a constant times the margin, B,d are the same in the two networks. Therefore, we compare the term $\sum_{i\in d} \frac{\|W_i\|_F}{\|W_i\|_2}$ for two networks. Calculating this value shows bigger generalization bound for RI-T and hence predicts worse generalization performance.



(a) CheXpert (b) clipart

Figure 33: Count of singular values smaller than a threshold